Just How Fair is an Unreactive World?

Srinivasan Raghuraman[*] Yibin Yang[†]

February 17, 2023

Abstract

Fitzi, Garay, Maurer, and Ostrovsky (J. Cryptology 2005) showed that in the presence of a dishonest majority, no primitive of cardinality $n-1$ is complete for realizing an arbitrary n-party functionality with *guaranteed output delivery*. In this work, we show that in the presence of $n-1$ corrupt parties, no *unreactive* primitive of cardinality $n-1$ is complete for realizing an arbitrary n-party functionality with *fairness*. We show more generally that for $t > \frac{n}{2}$, in the presence of t malicious parties, no *unreactive* primitive of cardinality t is complete for realizing an arbitrary n-party functionality with *fairness*. We complement this result by noting that $(t+1)$-wise fair exchange is complete for realizing an arbitrary n-party functionality with *fairness*. In order to prove our results, we utilize the primitive of fair coin tossing and introduce the notion of *predictability*, which we believe is of independent interest.

Keywords: Secure computation, unreactive functionalities, fair coin tossing.

[*]Visa Research, srraghur@visa.com.

[†]Georgia Institute of Technology, yyang811@gatech.edu. This work was done in part while the author was at Visa Research.

Contents

1 Introduction

Secure multiparty computation (MPC) [Yao86] allows a set of n mutually mistrusting parties to perform a joint computation on their inputs that reveals only the output of the computation and nothing else. Several definitions of MPC have been considered in the literature. Often, they have a lot to do with the kinds of adversaries we are trying to achieve security against, and in particular, the number of parties t that the adversary is allowed to corrupt. The most commonly used definition is that of *security-with-abort*, where the adversary is allowed to abort or quit after learning its output, even if the honest parties do not learn theirs. In contrast to security-with-abort, one can consider stronger notions of security such as *fairness* and *guaranteed output delivery*. Fairness means that either all parties get the output or none do. Guaranteed output delivery means that all parties get the output. In settings where a majority of the participating parties can be corrupted, that is, $t \geq \frac{n}{2}$, all feasibility results [Yao86, GMW87, BGW88, CCD88, RB89] that design a protocol for MPC only provide security-with-abort. On the other hand, when only up to $t < \frac{n}{3}$ parties can be corrupted, then there exist MPC protocols with guaranteed output delivery [BGW88, CCD88] (this result can be extended to a setting where up to $t < \frac{n}{2}$ parties can be corrupted assuming the existence of a broadcast channel [GMW87, RB89]). Cleve [Cle86] showed that dishonest majority fair coin tossing is impossible, inferring that MPC with even fairness is impossible in general for $t \geq \frac{n}{2}$ (although several works [GHKL11, GK09, Ash14, ABMO15] showed the existence of non-trivial functions for which MPC with fairness and even guaranteed output delivery is possible in the dishonest majority setting).

Given the above fairly tight characterization of what can be achieved in the realm of MPC, a natural question is whether additional resources, that we call *channels* or *functionalities*[1], help to achieve stronger security for MPC, and if so, by how much. Indeed, as we noted above, a broadcast channel moves the boundary from $t < \frac{n}{3}$ to $t < \frac{n}{2}$ for MPC with guaranteed output delivery. Even the impossibility result of Cleve [Cle86] can be trivially bypassed with access to a fair exchange functionality. One of the seminal works in this line is that of Fitzi, Garay, Maurer, and Ostrovsky [FGMO05] who studied functionalities that enable MPC with guaranteed output delivery in the presence of a dishonest majority. They showed that no functionality of cardinality[2] $n-1$ is complete for n-party MPC. More generally, for $n \geq 3$ and $\beta < n$, they show that no functionality of cardinality β is complete when $t \geq \lceil \frac{\beta-1}{\beta+1} \cdot n \rceil$. Also, when $t \geq n-2$, no functionality of cardinality $\beta < n$ is complete (they also show a primitive of cardinality n that is complete for n-party MPC when $t \geq n-2$).

The impossibility results in [FGMO05] are derived by showing the *impossibility of broadcast* given a functionality of cardinality β. Cohen and Lindell [CL17] showed that the presence of a broadcast channel is inconsequential to achieving the goal of fairness, that is, they showed that any protocol for fair computation that uses a broadcast channel can be compiled into one that does not use a broadcast channel assuming one-way functions (they also showed that assuming the existence of a broadcast channel, any protocol for fair secure computation can be compiled into one that provides guaranteed output delivery). Therefore, the impossibility results of [FGMO05] does not extend to MPC with fairness, giving rise to the question of whether there exist functionalities of cardinality $\beta < n$ that are complete for MPC with fairness.

Gordon et al. [GIM+10] propose primitives that are complete for MPC with fairness[3]. However,

[1] These channels may be implemented via a trusted third party, or hardware or cryptographic assumptions.

[2] Cardinality refers to the number of parties interacting with a single instance of the ideal primitive.

[3] In fact, some of their primitives are also complete for MPC with guaranteed output delivery. The upside of these

these primitives are of cardinality n, and thus do not answer the question of whether a primitive of cardinality less then n can be complete for MPC with fairness. Recently, Kumaresan et al. [KRS20] propose a functionality of cardinality 2 called *synchronizable fair exchange* ($\mathcal{F}_{\mathsf{SyX}}$) that is complete for MPC with fairness in the presence of a dishonest majority, thus answering the question raised above. However, $\mathcal{F}_{\mathsf{SyX}}$ is a *reactive* functionality. Reactive functionalities can be invoked multiple times and potentially maintain state between invocations. *Unreactive* functionalities, on the other hand, can only be invoked once. Reactive functionalities clearly have the potency to be far more powerful than unreactive ones. For this and other reasons, the assumption of having a reactive functionality is undoubtedly strong one and hard one to justify. Indeed, if one could achieve the same things that $\mathcal{F}_{\mathsf{SyX}}$ does, but with unreactive functionalities, that would be preferable. Given this, we pose the following question that we *completely* address in our work:

Just how fair is an unreactive world with only have unreactive functionalities?
Is MPC with fairness achievable with unreactive functionalities?

Our contributions. In this work, we show that an unreactive world is not very fair. On the negative side, we show that unreactive functionalities of cardinality β upper bounded by $n-1$ are incomplete for MPC with fairness. More generally, for $t > \frac{n}{2}$ and $\beta \leq t$, no unreactive functionality of cardinality β is complete for MPC with fairness. We establish this result by showing that a specific n-party primitive, *fair coin tossing*, cannot be realized using unreactive functionalities of cardinality t in the presence of t malicious parties for $t > \frac{n}{2}$.

One could view our work as an extension of the result of [Cle86] to the setting of unreactive functionalities. However, the extension is non-trivial as the techniques of [Cle86] face several challenges in the setting of unreactive functionalities. In order to surmount these challenges, we introduce the notion of *predictability* which we believe is of independent interest, as it provides highly tangible insight and directives in the design of protocols for fair coin tossing.

On the positive side, we show that for $t \geq \frac{n}{2}$[4] and $\beta = t+1$, the unreactive functionality of β-wise fair exchange is complete for MPC with fairness. For $t = \frac{n}{2}$ and $\beta = 2$, the unreactive functionality of 2-wise fair exchange is complete for MPC with fairness. This *entirely* covers the space of parameters for t and β.

We summarize our contributions in Table 1 and in the (informal) theorem below.

Theorem (informal).

- *For $\frac{n}{2} < t < n$, there does not exist a fair coin tossing protocol using unreactive primitives of cardinality upper bounded by t.*
- *For $\frac{n}{2} < t < n$, there exists a protocol for arbitrary MPC with fairness using $(t+1)$-wise fair exchange.*
- *For $t = \frac{n}{2}$, there exists a protocol for arbitrary MPC with fairness using 2-wise fair exchange.*

Note that our results have very interesting consequences. For instance, our results show that a 2-wise fair coin toss cannot be used to obtain a 3-wise fair coin toss in the presence of 2 malicious parties. Note that this is in contrast to the world of security-with-abort, where oblivious transfer or 2-wise MPC with abort can be used to obtain n-wise MPC with abort for all $n \geq 2$ [Kil88].

primitives is that unlike [FGMO05], their primitive complexity is independent of the function being computed.

[4]Note that for $t < \frac{n}{2}$, no functionality is needed for MPC with fairness.

Table 1: Our contributions.

t	**Insufficient functionalities for fair coin tossing**	**Sufficient functionalities for fair MPC**
$t < \frac{n}{2}$	–	Local computation [FGMvR02]
$t = \frac{n}{2}$	Local computation [Cle86]	2-wise fair exchange **[ours]**
$t > \frac{n}{2}$	Arbitrary unreactive t-wise **[ours]**	$(t+1)$-wise fair exchange[a] **[ours]**

[a]Or 2-wise $\mathcal{F}_{\mathsf{SyX}}$ (Lemma 5, [KRS20]).

Table 2: Positive results (IT is information-theoretic; Comp. is computational).

Protocol	**Assumption**	**Setting**	**Constraint**	**Bias**
[ABC+85]	OWF[a]	IT	$\perp$	$\Theta(\frac{1}{\sqrt{R}})$
[MNS09]	OT	Comp.	$n = 2$	$\Theta(\frac{1}{R})$
[BOO10]	OT	Comp.	$t < \frac{2}{3}n$	$\Theta(\frac{2^{2^{2t-n}}}{R})$
[HT14]	OT	Comp.	$n = 3$	$\mathcal{O}(\frac{\log^3 R}{R})$
[AO16]	OT	Comp.	$t < \frac{3}{4}n$	$\mathcal{O}(\frac{2^{2^n}\log^3 R}{R})$
[BHLT17]	OT	Comp.	$n \leq \frac{1}{2}\log\log R$	$\mathcal{O}(\frac{n^4 \cdot 2^n \cdot \sqrt{\log R}}{R^{1/2+1/(2^{n-1}-2)}})$
[ours]	2-wise $\mathcal{F}_{\mathsf{X}}$[b]	IT	$t = \frac{n}{2}$	0
[ours]	$(t+1)$-wise $\mathcal{F}_{\mathsf{X}}$	IT	$\perp$	0

[a]OWF denotes one-way function.
[b]$\mathcal{F}_{\mathsf{X}}$ denotes fair exchange.

1.1 Related Work on Coin Tossing

Cleve [Cle86] showed that for any n-party R-round coin tossing protocol where parties are connected with k communication channels, there exists an adversary that can bias the honest parties' common output bit by $\Theta(\frac{1}{R})$[5]. Prior to [Cle86]'s lower bound, Awerbuch et al. [ABC+85] designed a coin tossing protocol with $\Theta(\frac{1}{\sqrt{R}})$[6] bias[7]. [ABC+85]'s protocol works under *any* hardness assumption and for *any* number of parties. Since then, there have been numerous works that focus on eliminating this gap between $\Theta(\frac{1}{\sqrt{R}})$ for the protocols and $\Theta(\frac{1}{R})$ in the lower bound.

Many works (e.g., [MNS09, BOO10, HT14, AO16, BHLT17]) have tried to design new coin tossing protocols to get as close to [Cle86]'s bound as possible, while others (e.g., [MW20, BHMO22]) tried to prove a tighter bound. In the positive direction, the setting is computational and often assumes the existence of *oblivious transfer* (OT). We summarize the positive and negative results from the literature along with our own results in Table 2 and Table 3 respectively. Despite the array of works on the topic, the problem of designing a coin tossing protocol in the computational setting that achieves [Cle86]'s lower bound in general still remains open.

Other lines of work focus on coin tossing protocols or even general multi-party computations with weaker security guarantees. [Blu83, IL89, Nao91, HNO+09, MPS10, HO14, BHT18] studied

[5]More explicitly, the bias is $\Theta(\frac{1}{nRk})$.
[6]More explicitly, the bias is $\Theta(\frac{n}{\sqrt{R}})$.
[7][Cle86] specified [ABC+85]'s protocol for the 2-party case and analyzed the bias.

Table 3: Negative results (IT is information-theoretic; Comp. is computational).

Work	Assumption	Setting	Constraint	Bias
[Cle86]	$\perp$	IT/Comp.	$t > \frac{n}{2}$	$\Theta(\frac{1}{R})$
[CI93]	$\perp$	IT	$t > \frac{n}{2}$	$\Theta(\frac{1}{\sqrt{R}})$
[MW20]	Black-box OWF, RO[a]	IT[b]	$n = 2, t = 1$[c]	$\Theta(\frac{1}{\sqrt{R}})$
[BHMO22]	$\perp$	Comp.	$\exists k \in \mathbb{N}, n^k \geq R$[d]	$\Theta(\frac{1}{\sqrt{R}\log(R)^k})$
[ours]	Unreactive t-wise $\mathcal{F}$s	Comp.	$t > \frac{n}{2}$	$\Theta(\frac{1}{R})$

[a] RO denotes random oracle.

[b] However, the adversary only allows to make polynomially-many additional queries to the random oracle.

[c] Can be extended to $t > \lfloor \frac{n}{2} \rfloor$ with an adjusted bound.

[d] Can be extended to $t > \lfloor \frac{n}{2} \rfloor$ with an adjusted bound.

coin tossing protocols with security-with-abort. [BK14, KVV16] studied general MPC protocols in the model of *fairness with penalty*, where the adversary must pay a penalty (e.g., via digital currency such as Bitcoin [Nak08]) if it aborts after learning the output. Very recently, [CGL$^+$18, WAS22] studied coin tossing protocols in a game-theoretic sense where each party would like to bias the output of the coin toss in a specific direction. Coin tossing protocols have been proposed in several other models as well. [BOL85, Sak89, AN93, Fei99, RZ01, GKP15, HKH20, KKR21] studied *collective coin-flipping* in the *full information model* where the parties are connected to a broadcast channel and keep no private state between the different communication rounds. [BC90, ATSVY00, Amb01, ABDR04] studied the quantum coin tossing protocols. [MN05] studied the use of tamper-evident seals in coin tossing protocols.

1.2 Technical Overview

Upper bounds. We note that fair MPC can be reduced to fair reconstruction of a secret [GIM$^+$10, KRS20]. We simply demonstrate that a $(t + 1)$-wise fair exchange suffices to perform fair reconstruction in the presence of t malicious parties. Intuitively, this follows from the fact that there is always an honest party who is part of the $(t + 1)$-wise fair exchange and hence learns the result of the exchange if the adversary does. Our more interesting upper bound is for the case of $t = \frac{n}{2}$ where 2-wise fair exchange suffices. The intuition of the protocol that achieves this is the following. Let s denote the secret to be reconstructed. We secret share s using an $\left(\frac{n}{2} + 1\right)$-out-of-$n$ secret sharing. Our protocol will require all pairs of parties to exchange their shares using 2-wise fair exchanges. Notice that both the adversary and the honest parties are one share away from the output. Thus, for anyone to learn the output, at least one pair of parties, one of whom is honest and the other malicious, must perform a 2-wise exchange successfully. But on doing so, both sides have enough shares to reconstruct s.

Lower bounds. A first attempt at proving our lower bounds would be to consider the lower bounds of [Cle86] and somehow generalize them to a model that allows the use unreactive functionalities of cardinality at least 2. We call this model an *unreactive world*, which we formally define in Section 2.6. However, this exercise turns out to be a futile one. Right off the bat, for some parameters, fair MPC is not achievable in [Cle86]'s model, but is achievable in ours, as shown by our upper bounds (Cf. Section 3). Thus, one needs a different approach to prove lower bounds in unreactive worlds.

To this end, we introduce the notions of *predictability* and a *predictor* for coin tossing protocols (Cf. Definitions 6 and 8).[8] At a high level, a predictor is some computation which allows parties to calculate the final output of a coin tossing protocol after executing just a prefix of it. A very intuitive understanding of predictability is the following. Consider the beginning of the protocol, where parties have access to their local state[9] and nothing else. It is easy to see that the set of all parties can jointly predict the output of the protocol at this stage, while individual parties may not, in fact, should not, be able to. However, at the end of the protocol, each individual party is able to predict the final output. Thus, over the course of protocol, predictability of every subset of parties evolves. Our proof demonstrates and exploits the fact that there are points in the protocol where some subsets of parties can predict the output non-negligibly better than others. In our proof, we call such a non-negligible difference in predictabilities as a *gap*. The crucial step of our proofs will be to locate a gap that will help us construct adversarial strategies that will bias the output of an honest party non-negligibly. We call such a gap an *attackable gap*.

To locate an attackable gap, we introduce the notion of a *predictor graph*. The vertices of this graph are predictors. Two predictors are connected by an edge in the predictor graph if and only they form what we call an *attackable pair* (Cf. Definitions 7 and 9). It turns out that a non-negligible gap in the predictabilities of an attackable pair is an attackable gap. Thus, it suffices to find an attackable pair with a non-negligible gap. By virtue of the triangle inequality, this reduces to the following problem: find a path of polynomial length in the predictor graph whose endpoints are predictors with a non-negligible gap. Our entire proof technique is demonstrating how to find such paths and thus locate an attackable gap.

Organization of the work. Due to space limitations, we defer some proofs of lemmas as supplementary material. The work is organized as follows:

- Section 2 contains preliminaries on notation and our model called *unreactive worlds*.
- In Section 3, we present our positive results to show how we can bypass Cleve's lower bound in unreactive worlds.
- In Section 4, we explain how we prove our lower bound in the two-party unreactive worlds using the notion of *predictability* and compare our methodology with [Cle86].
- In Section 5, we explain how we prove our lower bound in the multi-party unreactive worlds in the presence of *all-but-one* corruptions.
- In Section 6, we explain how we prove our lower bound in the multi-party unreactive worlds in the presence of arbitrary *threshold* corruptions.
- In Section 7, we tackle some assumptions we make in the paper for ease of presentation and refine our results to overcome these assumptions.

[8] Our notions of predictors and predictabilities are distinct from other notions (implictly) considered in prior works (e.g. [HZ10, IKK+11, HIK+19]).

[9] Let us assume that the local state contains all the randomness that the party will ever use through the course of the protocol.

2 Preliminaries

2.1 Notation and Definitions

For $n \in \mathbb{N}$, let $[n] = \{1, 2, \ldots, n\}$. Let $\lambda \in \mathbb{N}$ denote the security parameter. Symbols in with an arrow over them such as $\overrightarrow{a}$ denote vectors. By a_i we denote the i-th element of the vector $\overrightarrow{a}$. By $\mathrm{poly}(\cdot)$, we denote any function which is bounded by a polynomial in its argument. An algorithm $\mathcal{T}$ is said to be PPT if it is modeled as a probabilistic Turing machine that runs in time polynomial in λ. Informally, we say that a function is negligible, denoted by $\delta(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ. Similarly, we denote a function is non-negligible as $\epsilon(\lambda)$.

Let $\mathcal{X}, \mathcal{Y}$ be two probability distributions over some set S. Their *statistical distance* is

$$\mathbf{SD}\left(\mathcal{X}, \mathcal{Y}\right) \stackrel{\text{def}}{=} \max_{T \subseteq S} \left\{ |\Pr[\mathcal{X} \in T] - \Pr[\mathcal{Y} \in T]| \right\}$$

We say that $\mathcal{X}$ and $\mathcal{Y}$ are ν-close if $\mathbf{SD}\left(\mathcal{X}, \mathcal{Y}\right) \leq \nu$ denoted by $\mathcal{X} \approx_\nu \mathcal{Y}$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are identical if $\mathbf{SD}\left(\mathcal{X}, \mathcal{Y}\right) = 0$ denoted by $\mathcal{X} \equiv \mathcal{Y}$.

2.2 Secure Computation

We recall most of the definitions regarding secure computation from [CL17] and [GHKL11]. We present them here for the sake of completeness and self-containedness. Consider the scenario of n parties $P_1, \ldots, P_n$ with private inputs $x_1, \ldots, x_n \in \mathcal{X}$.

2.2.1 Functionalities

A functionality f is a randomized process that maps n-tuples of inputs to n-tuples of outputs, that is, $f : \mathcal{X}^n \to \mathcal{Y}^n$. We write $f = (f^1, \ldots, f^n)$ if we wish to emphasize the n outputs of f, but stress that if $f^1, \ldots, f^n$ are randomized, then the outputs of $f^1, \ldots, f^n$ are correlated random variables. Here, we refer to n as the *cardinality* of the functionality f.

2.2.2 Adversaries

We consider security against *static t-threshold adversaries*, that is, adversaries that corrupt a set of at most t parties, where $0 \leq t < n$[10]. We assume the adversary to be malicious. That is, the corrupted parties may deviate arbitrarily from an assigned protocol.

2.2.3 Model

We assume the parties are connected via a fully connected point-to-point network; we refer to this model as the `point-to-point` model. We sometimes assume that the parties are given access to a physical broadcast channel (defined in Section 2.8)[11] in addition to the point-to-point network; we refer to this model as the `broadcast` model. The communication lines between parties are assumed to be ideally authenticated and private (and thus an adversary cannot read or modify messages sent between two honest parties). Furthermore, the delivery of messages between honest parties is guaranteed.

[10] Note that when $t = n$, there is nothing to prove.

[11] This can also be viewed as working in the $\mathcal{F}_{\mathsf{bc}}$-hybrid model. See Section 2.4.

2.2.4 Protocol

An n-party protocol for computing a functionality f is a protocol running in polynomial time and satisfying the following functional requirement: if for every $i \in [n]$, party P_i begins with private input $x_i \in \mathcal{X}$, then the joint distribution of the outputs of the parties is statistically close to $(f^1(\overrightarrow{x}), \ldots, f^n(\overrightarrow{x}))$. We assume that the protocol is executed in a synchronous network, that is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their message for this round) followed by a *receive* phase (where they receive messages from other parties). The adversary, being malicious, is also *rushing* which means that it can see the messages the honest parties send in a round, before determining the messages that the corrupted parties send in that round.

The security of a protocol is analyzed by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an *ideal* computation involving an incorruptible *trusted party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted party exists) can do no more harm than if it were involved in the above-described ideal computation.

2.2.5 Security with Guaranteed Output Delivery

The security of a protocol is analyzed by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an *ideal* computation involving an incorruptible *trusted party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted party exists) can do no more harm than if it were involved in the above-described ideal computation.

Execution in the ideal model. The parties are $P_1, \ldots, P_n$, and there is an adversary $\mathcal{A}$ who has corrupted at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. An ideal execution for the computation of f proceeds as follows:

- **Inputs:** $P_1, \ldots, P_n$ hold their private inputs $x_1, \ldots, x_n \in \mathcal{X}$; the adversary $\mathcal{A}$ receives an auxiliary input z.

- **Send inputs to trusted party:** The honest parties send their inputs to the trusted party. The corrupted parties controlled by $\mathcal{A}$ may send any values of their choice. Denote the inputs sent to the trusted party by $x'_1, \ldots, x'_n$.

- **Trusted party sends outputs:** If $x'_i \notin \mathcal{X}$ for any $i \in [n]$, the trusted party sets x'_i to some default input in $\mathcal{X}$. Then, the trusted party chooses r uniformly at random and sends $f^i(x'_1, \ldots, x'_n; r)$ to party P_i for every $i \in [n]$.

- **Outputs:** The honest parties output whatever was sent by the trusted party. The corrupted parties output nothing and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\textsc{Ideal}^{\mathsf{g.d.}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model described above.

Execution in the real model. We next consider the real model in which an n-party protocol π is executed by $P_1,\ldots,P_n$ (and there is no trusted party). In this case, the adversary $\mathcal{A}$ gets the inputs of the corrupted party and sends all messages on behalf of these parties, using an arbitrary polynomial-time strategy. The honest parties follow the instructions of π.

Let f be as above and let π be an n-party protocol computing f. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input z. We let $\textsc{Real}_{\pi,\mathcal{I},\mathcal{A}(z)}(x_1,\ldots,x_n,\lambda)$ be the random variable consisting of the view of the adversary and the output of the honest parties following an execution of π where P_i begins by holding x_i for every $i \in [n]$.

Security as emulation of an ideal execution in the real model. Having defined the ideal and real models, we can now define security of a protocol. Loosely speaking, the definition asserts that a secure protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated as follows.

Definition 1. *Protocol π is said to securely compute f with guaranteed output delivery if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \leq t$,*

$$\left\{\textsc{Ideal}^{\mathsf{g.d.}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)\right\}_{\overrightarrow{x}\in\mathcal{X}^n,z\in\{0,1\}^*} \equiv \left\{\textsc{Real}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)\right\}_{\overrightarrow{x}\in\mathcal{X}^n,z\in\{0,1\}^*}$$

We will sometimes relax security to statistical or computational definitions. A protocol is statistically secure if the random variables $\textsc{Ideal}^{\mathsf{g.d.}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)$ and $\textsc{Real}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)$ are statistically close, and computationally secure if they are computationally indistinguishable.

2.2.6 Security with Fairness

In this definition, the execution of the protocol can terminate in two possible ways: the first is when all parties receive their prescribed output (as in the case of guaranteed output delivery) and the second is when all parties (including the corrupted parties) abort without receiving output. The only change from the definition in the case of guaranteed output delivery above is with regard to the ideal model for computing f, which is now defined as follows:

Execution in the ideal model. The parties are $P_1,\ldots,P_n$, and there is an adversary $\mathcal{A}$ who has corrupted at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. An ideal execution for the computation of f proceeds as follows:

- **Inputs:** $P_1,\ldots,P_n$ hold their private inputs $x_1,\ldots,x_n \in \mathcal{X}$; the adversary $\mathcal{A}$ receives an auxiliary input z.
- **Send inputs to trusted party:** The honest parties send their inputs to the trusted party. The corrupted parties controlled by $\mathcal{A}$ may send any values of their choice. In addition, there exists a special abort input. Denote the inputs sent to the trusted party by $x'_1,\ldots,x'_n$.

- **Trusted party sends outputs:** If $x'_i \notin \mathcal{X}$ for any $i \in [n]$, the trusted party sets x'_i to some default input in $\mathcal{X}$. If there exists an $i \in [n]$ such that $x'_i = \mathsf{abort}$, the trusted party sends $\perp$ to all the parties. Otherwise, the trusted party chooses r uniformly at random, computes $z_i = f^i(x'_1, \ldots, x'_n; r)$ for every $i \in [n]$ and sends z_i to P_i for every $i \in [n]$.
- **Outputs:** The honest parties output whatever was sent by the trusted party. The corrupted parties output nothing and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\text{IDEAL}^{\mathsf{fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model described above.

Definition 2. *Protocol π is said to securely compute f with fairness if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \leq t$,*

$$\left\{\text{IDEAL}^{\mathsf{fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*} \equiv \left\{\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*}$$

We will sometimes relax security to statistical or computational definitions. A protocol is statistically secure if the random variables $\text{IDEAL}^{\mathsf{fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)$ and $\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)$ are statistically close, and computationally secure if they are computationally indistinguishable.

2.3 Coin Tossing Protocols

In this section, we formally define coin tossing protocols. While we follow prior works, we present the definitions for the sake of completeness.

Definition 3 (Coin Tossing Protocols)**.** *Consider a protocol π among $n \in \mathbb{N}$ parties $P_1, \ldots, P_n$ where each party P_i takes as input the string 1^λ and outputs a single bit $res_i \in \{0, 1\}$ after the execution of π. The protocol π is said to be a* coin tossing protocol *if and only if when all parties follow the protocol:*

- ***[Uniform Coin]*** *For all $i \in [n]$,*

$$\left|\Pr[res_i = 0] - \frac{1}{2}\right| \leq \delta_i(\lambda)$$

  for some negligible function $\delta_i(\lambda)$.

- ***[Agreement]*** *For any $i, j \in [n]$,*

$$\Pr[res_i = res_j] = 1$$

Given the above, throughout our work, we will denote the output of a coin tossing protocol by res, as opposed to considering res_i for each $P_{i \in [n]}$.

The security property called t-resistance of coin tossing protocols is that even if $t \in [n-1]$ parties in the n-party coin tossing protocol π are corrupted and deviate arbitrarily from the protocol, the remaining $n - t$ honest parties each agree on and output a uniform bit.

Definition 4 (t-resistance)**.** *Consider a coin tossing protocol π among $n \in \mathbb{N}$ parties $P_1, \ldots, P_n$. The protocol π is said to be t-resistant (where $t \in [n-1]$) if and only if when any t parties are corrupt and deviate arbitrarily from π, and the remaining $n-t$ honest parties execute π:*

- ***[Uniform Coin]*** *For any output $\widetilde{res}_i$ of honest P_i where $i \in [n]$*

$$\left|\Pr[\widetilde{res}_i = 0] - \frac{1}{2}\right| \leq \delta_i(\lambda)$$

  for some negligible function $\delta_i(\lambda)$.

- ***[Agreement]*** *For any $i, j \in [n]$ such that P_i and P_j are honest,*

$$\Pr[\widetilde{res}_i = \widetilde{res}_j] = 1$$

Throughout our work, if an n-party coin tossing protocol satisfies t-resistance, we call it an n-party t-fair coin tossing protocol. We may omit t and n if it is clear from context.

In Definitions 3 and 4, we ask that honest parties make *perfect agreement*, that is, the probability that they agree is 1. This is merely for ease of presentation. All of our results also apply to the setting where honest parties only make statistical agreement, that is, they agree with all but probability negligible in λ. We elaborate on this in Section 7.

A critical notion, introduced by [Cle86], is that of *bias*, which is a measure of the non-uniformity (with respect to a uniform coin) of the honest party's coin in the presence of an adversary.

Definition 5 (Bias)**.** *Consider a coin tossing protocol π among $n \in \mathbb{N}$ parties $P_1, \ldots, P_n$. We say an adversary (set of corrupt parties) can bias an honest party P_k (for $k \in [n]$) by $\nu(\lambda)$ if and only if the output $\widetilde{res}_k$ of P_k satisfy*

$$\left|\Pr[\widetilde{res}_k = 0] - \frac{1}{2}\right| \geq \nu(\lambda)$$

2.4 The Hybrid Model

We recall the definition of the hybrid model from [GHKL11] and [CL17]. The hybrid model combines both the real and ideal worlds. Specifically, an execution of a protocol π in the $\mathcal{G}$-hybrid model, for some functionality $\mathcal{G}$, involves parties sending normal messages to each other (as in the real model) and, in addition, having access to a trusted party computing $\mathcal{G}$. The parties communicate with this trusted party in exactly the same way as in the ideal models described above; the question of which ideal model is taken must be specified.

Let $\mathsf{type} \in \{\mathsf{g.d.}, \mathsf{fair}\}$. Let $\mathcal{G}$ be a functionality and let π be an n-party protocol for computing some functionality f, where π includes real messages between the parties as well as calls to $\mathcal{G}$. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input z. $\mathcal{A}$ corrupts at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. Let $\textsc{Hybrid}^{\mathcal{G},\mathsf{type}}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)$ be the random variable consisting of the view of the adversary and the output of the honest parties, following an execution of π with ideal calls to a trusted party computing $\mathcal{G}$ according to the ideal model "type" where P_i begins by holding x_i for every $i \in [n]$. Security in the model "type" can be defined via natural modifications of Definitions 1 and 2. We call this the $(\mathcal{G}, \mathsf{type})$-hybrid model.

The hybrid model gives a powerful tool for proving the security of protocols. Specifically, we may design a real-world protocol for securely computing some functionality f by first constructing a protocol for computing f in the $\mathcal{G}$-hybrid model. Letting π denote the protocol thus constructed (in the $\mathcal{G}$-hybrid model), we denote by π^ρ the real-world protocol in which calls to $\mathcal{G}$ are replaced by sequential execution of a real-world protocol ρ that computes $\mathcal{G}$ in the ideal model "type". "Sequential" here implies that only one execution of ρ is carried out at any time, and no other π-protocol messages are sent during the execution of ρ. The results of [Can00] then imply that if π securely computes f in the $(\mathcal{G}, \mathsf{type})$-hybrid model, and ρ securely computes $\mathcal{G}$, then the composed protocol π^ρ securely computes f (in the real world). For completeness, we state this result formally as we will use it in this work.

Lemma 1. *Let* $\mathsf{type}_1, \mathsf{type}_2 \in \{\mathsf{g.d.}, \mathsf{fair}\}$. *Let* $\mathcal{G}$ *be an* n*-party functionality. Let* ρ *be a protocol that securely computes* $\mathcal{G}$ *with* type_1*, and let* π *be a protocol that securely computes* f *with* type_2 *in the* $(\mathcal{G}, \mathsf{type}_1)$*-hybrid model. Then protocol* π^ρ *securely computes* f *with* type_2 *in the real model.*

Sometimes, while working in a hybrid model, say the $(\mathcal{G}, \mathsf{type})$-hybrid model, we will suppress type and simply state that we are working in the $\mathcal{G}$-hybrid model. This is because type is implied by the context, $\mathcal{G}$. For instance, unless specified otherwise, when $\mathcal{G} = \mathcal{F}_{\mathsf{bc}}$[12] (broadcast functionality), $\mathsf{type} = \mathsf{g.d.}$.

When working in a hybrid model that uses multiple ideal functionalities, $\mathcal{G}_1, \ldots, \mathcal{G}_k$ with associated types $\mathsf{type}_1, \ldots, \mathsf{type}_k$ for some $k \in \mathbb{N}$, we call it the $(\mathcal{G}_1, \mathsf{type}_1, \ldots, \mathcal{G}_k, \mathsf{type}_k)$-hybrid model. Furthermore, we will suppress type_j when type_j is implied by the context, $\mathcal{G}_j$ for $j \in [k]$.

2.5 Unreactive Functionalities

Consider a functionality $\mathcal{G}$ with an associated type type. We say that $\mathcal{G}$ is unreactive if and only if the computation performed by $\mathcal{G}$ can be realized by a circuit with access to randomness[13]. An alternative characterization would be that $\mathcal{G}$ only has one phase and hence can be emulated by a one-time[14] trusted third-party with access to randomness. Therefore, an instance of $\mathcal{G}$ will (1) receive the inputs; (2) compute the intended program to obtain the outputs; and (3) deliver the outputs (according to the type). After the delivery process, the instance will be totally obliterated, i.e., it can no longer be accessed. This would mean that different invocations of the same functionality $\mathcal{G}$ can be seen as invocations of multiple "different" functionalities.

2.6 Our Model: The Unreactive World

Let $n, \beta \in \mathbb{N}$ and $\beta < n$. Our (n, β)-*unreactive world* is a hybrid model where n parties $\mathbb{P} \triangleq \{P_1, \ldots, P_n\}$ are equipped with:

1. A broadcast channel, in which any of the parties can act as the broadcaster.
2. An arbitrary number of arbitrary unreactive functionalities of type $\mathsf{type} = \mathsf{g.d.}$ whose cardinality is upper bounded by β.

[12] See Section 2.8.

[13] One way to model this is to consider circuits which in addition to regular computational gates, additionally have "random" gates that simply produce random bits as output.

[14] No internal state is retained between invocations of the functionality.

Protocols in (n, β)-unreactive world adhere to a specific syntax which called the (n, R, β)-*unreactive syntax*. For simplicity, we first consider the case of $\beta = n - 1$. The $(n, R, n-1)$-unreactive syntax describes an R-round protocol that takes the following form:

- Each P_i generates its own local random tape r_i.
- For each round $k \in [R]$, there are n unreactive functionalities of cardinality $n-1$ being executed *in sequence* denoted by $\mathcal{F}^{(k)}_{\mathbb{P}\setminus\{P_1\}}, \mathcal{F}^{(k)}_{\mathbb{P}\setminus\{P_2\}}, \ldots, \mathcal{F}^{(k)}_{\mathbb{P}\setminus\{P_n\}}$. $\mathcal{F}^{(k)}_{\mathbb{S}}$ denotes the unreactive functionality which is connected to the set of parties $\mathbb{S}$ in the round k.
- After the unreactive functionality phase, each round will also contain a broadcast phase, where each party (from P_1 to P_n) broadcasts in order.
- Finally, each party P_i calculates the output by invoking a procedure Π_i on its random tape r_i and the $2R(n-1)$ messages it obtained in the protocol.

We describe the above $(n, R, n-1)$-unreactive syntax algorithmically in Figure 1.

We generalize the $(n, R, n-1)$-unreactive syntax to the (n, R, β)-unreactive syntax arbitrary $\beta < n$ as follows. The main difference lies in the unreactive functionality phase of each round, where there are $\binom{n}{\beta}$ unreactive functionalities of cardinality β being executed in sequence. These $\binom{n}{\beta}$ unreactive functionalities are connected to the $\binom{n}{\beta}$ different β-sized subsets of the n parties. Finally, each party P_i calculates the output by invoking a procedure Π_i on its random tape r_i and the $R(\binom{n-1}{\beta-1} + (n-1))$ messages it obtained in the protocol.

Note that the (n, R, β)-unreactive syntax supports unreactive functionalities of cardinality below β as well. Indeed, a functionlality of cardinality k can be emulated by some functionality of cardinality k' for any $k \leq k'$.

2.6.1 Generality

In this section, we argue that our unreactive syntax for protocols captures all protocols that are designed in the corresponding unreactive world. We do this by going over the ways in which a generic protocol in the unreactive world can differ in syntax from the one we have outlined, and arguing that such a protocol can be turned into one that follows our syntax. Indeed, such a transformation may incur a change in number of rounds (or more precisely, steps). We describe our *two* transformation steps, in order, ahead.

Serialization. In our syntax, parties perform invocations serially. A generic protocol in our model may have steps where multiple parties invoke different functionalities at the same time. For instance, a generic protocol may have a step where parties P_1 and P_2 are expected to broadcast certain messages at the same time. For all such "parallel" invocations, we simply *arbitrarily* serialize the "parallel" invocations. We argue that if the generic protocol was secure, then so is our serialized protocol. We argue this by noting that if there is an adversary $\mathcal{A}_{\mathsf{serial}}$ that can break the security of the underlying protocol while executing the serialized protocol, then, there exists an adversary $\mathcal{A}_{\mathsf{generic}}$ that does the same while executing the generic protocol. $\mathcal{A}_{\mathsf{generic}}$ does exactly what $\mathcal{A}_{\mathsf{serial}}$ does. The salient point to make note of here is that $\mathcal{A}_{\mathsf{generic}}$ can mimic $\mathcal{A}_{\mathsf{serial}}$, despite the generic protocol having parallel invocations since $\mathcal{A}_{\mathsf{generic}}$ is *rushing*. This means that if $\mathcal{A}_{\mathsf{serial}}$ was launching an attack in a step that involved parallel invocations in the generic protocol that

Algorithm $\pi_k(inputs, 1^\lambda)^{\mathcal{F}_{\mathsf{bc}}, \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_1\}}, \ldots, \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_{k-1}\}}, \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_{k+1}\}}, \ldots, \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_n\}}, \ldots, \mathcal{F}^{(R)}_{\mathbb{P}\setminus\{P_n\}}}$

$r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$

/* For round 1 */
$in_1 \leftarrow \Pi_{k,1}(r, inputs)$
$out_1 \leftarrow \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_1\}}(\cdot, \ldots, \cdot, in_1, \cdot, \ldots, \cdot)$
$in_2 \leftarrow \Pi_{k,2}(r, out_1, inputs)$
$out_2 \leftarrow \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_2\}}(\cdot, \ldots, \cdot, in_2, \cdot, \ldots, \cdot)$
...
$in_{k-1} \leftarrow \Pi_{k,k-1}(r, out_1, out_2, \ldots, out_{k-2}, inputs)$
$out_{k-1} \leftarrow \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_{k-1}\}}(\cdot, \ldots, \cdot, in_{k-1}, \cdot, \ldots, \cdot)$
$in_k \leftarrow \Pi_{k,k}(r, out_1, out_2, \ldots, out_{k-1}, inputs)$
$out_k \leftarrow \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_{k+1}\}}(\cdot, \ldots, \cdot, in_k, \cdot, \ldots, \cdot)$
...
$in_{n-1} \leftarrow \Pi_{k,n-1}(r, out_1, out_2, \ldots, out_{n-2}, inputs)$
$out_{n-1} \leftarrow \mathcal{F}^{(1)}_{\mathbb{P}\setminus\{P_n\}}(\cdot, \ldots, \cdot, in_{n-1}, \cdot, \ldots, \cdot)$

let $br_1 \triangleq \{\}$
receive $br_{1,1}$, append $br_{1,1}$ to br_1
receive $br_{1,2}$, append $br_{1,2}$ to br_1
...
receive $br_{1,k-1}$, append $br_{1,k-1}$ to br_1
$br_{1,k} \leftarrow BR_{k,1}(r, br_1, out_1, out_2, \ldots, out_{n-1}, inputs)$
$\mathcal{F}_{\mathsf{bc}}(br_{1,k})$, append $br_{1,k}$ to br_1
receive $br_{1,k+1}$, append $br_{1,k+1}$ to br_1
...
receive $br_{1,n}$, append $br_{1,n}$ to br_1

...

/* For round i */
$in_{(i-1)(n-1)+1} \leftarrow \Pi_{k,(i-1)(n-1)+1}(r, br_1, \ldots, br_{i-1}, out_1, \ldots, out_{(i-1)(n-1)}, inputs)$
$out_{(i-1)(n-1)+1} \leftarrow \mathcal{F}^{(i)}_{\mathbb{P}\setminus\{P_1\}}(\cdot, \ldots, \cdot, in_{(i-1)(n-1)+1}, \cdot, \ldots, \cdot)$
...
$in_{(i-1)(n-1)+n-1} \leftarrow \Pi_{k,(i-1)(n-1)+n-1}(r, br_1, \ldots, br_{i-1}, out_1, \ldots, out_{(i-1)(n-1)+n-2}, inputs)$
$out_{(i-1)(n-1)+n-1} \leftarrow \mathcal{F}^{(i)}_{\mathbb{P}\setminus\{P_n\}}(\cdot, \ldots, \cdot, in_{(i-1)(n-1)+n-1}, \cdot, \ldots, \cdot)$
let $br_i \triangleq \{\}$
receive $br_{i,1}$, append $br_{i,1}$ to br_i
...
receive $br_{i,k-1}$, append $br_{i,k-1}$ to br_i
$br_{i,k} \leftarrow BR_{k,i}(r, br_1, \ldots, br_{i-1}, out_1, \ldots, out_{(i-1)(n-1)+n-1}, inputs)$
$\mathcal{F}_{\mathsf{bc}}(br_{i,k})$, append $br_{i,k}$ to br_i
...
receive $br_{i,n}$, append $br_{i,n}$ to br_i

...

/* After R rounds */
$res_k \leftarrow \Pi_k(r, br_1, \ldots, br_R, out_1, \ldots, out_{R(n-1)}, inputs)$

return res_k

Figure 1: The $(n, R, n-1)$-unreactive Syntax

we serialized, $\mathcal{A}_{\mathsf{generic}}$, being a rushing adversary, could wait for the honest parties to execute their parallel invocations before executing its own, and this would exactly emulate the serial invocations performed before that point in the serialized protocol. It is immediate that this transformation would make no change to the number of steps in the protocol.

Sequentialization. In our syntax, parties invoke the unreactive functionalities first, in sequence, and then broadcast, in sequence. A generic protocol in our model may not follow the same order of operations. For instance, a generic protocol may begin with party P_2 broadcasting a certain message. For all such mismatches, we simply "pad" or "line" the generic protocol with "dummy" invocations of functionalities, both unreactive (the functionality itself can be set to be a "dummy" functionality that ignores inputs and produces no outputs) and broadcast (the message broadcast would be a "dummy" message). Since every step in the generic protocol could at most end up being its own round in our syntax, and a round in our syntax involves $\mathcal{O}(\binom{n}{\beta})$ steps, this transformation would blow up the number of steps in the protocol by a factor of at most $\mathcal{O}(\binom{n}{\beta})$ in the (n, β)-unreactive world.

Transforming a generic protocol. From the explanation above, it is clear that if one were to take a generic protocol in our model, serialize it and then sequentialize that, the resulting protocol would exactly follow our syntax. The resulting protocol would have a number of steps blown up by a factor of at most $\mathcal{O}(\binom{n}{\beta})$.

Steps vs. Rounds. While we have discussed the impact of the transformation on the number of steps, we haven't talked about the number of rounds of the transformed protocol. For that, we first note that what we call rounds in our syntax does in fact translate to something more akin to a *phase*, since each "round", or phase, contains $\mathcal{O}(\binom{n}{\beta})$ serial steps. If one were to consider every serial step to be a round, then our transformed protocol would have a number of steps blown up by a factor of at most $\mathcal{O}(\binom{n}{\beta}\gamma)$, where γ denotes the maximum parallel invocations performed by the generic protocol in any given step. We state this as a lemma below.

Lemma 2. *Let $n, \beta \in \mathbb{N}$ and $\beta < n$. Given an R-round protocol among n parties in the (n, β)-unreactive world that makes at most γ parallel invocations to any of the functionalities in any given step, there exists an $R \cdot \mathcal{O}(\binom{n}{\beta}\gamma)$-round protocol among n parties that follows our $(n, R \cdot \mathcal{O}(\binom{n}{\beta}\gamma), \beta)$-unreactive syntax.*

Remark. We note that [Cle86] implicitly assumes serialization and sequentialization as well. While not explicit, these steps are indeed needed to show that the syntax in [Cle86] does capture all protocols in that model.

2.7 Fairness versus Guaranteed Output Delivery

We recall here some of the results from [CL17].

Lemma 3. *[CL17] Consider n parties $P_1, \ldots, P_n$ in a model with a broadcast channel. Then, assuming the existence of one-way functions, for any functionality $f : \mathcal{X}^n \to \mathcal{Y}^n$, if there exists a protocol π which securely computes f with fairness, then there exists a protocol π' which securely computes f with guaranteed output delivery.*

Preliminaries: $I \subseteq [n]$. The functionality proceeds as follows:

- Sample a uniform bit $b \leftarrow \{0, 1\}$.
- Send b to all parties P_i for $i \in I$.

Figure 2: The ideal functionality $\mathcal{F}_{\mathsf{coin}}$.

Preliminaries: $x \in \{0, 1\}^*$. The functionality proceeds as follows:

- Upon receiving the input x from the sender P_1, send x to all parties $P_1, \ldots, P_n$.

Figure 3: The ideal functionality $\mathcal{F}_{\mathsf{bc}}$.

Lemma 4. *[CL17] Consider n parties $P_1, \ldots, P_n$ in a model with a broadcast channel. Then, assuming the existence of one-way functions, for any functionality $f : \mathcal{X}^n \to \mathcal{Y}^n$, if there exists a protocol π which securely computes f with fairness, then there exists a protocol π' which securely computes f with fairness and does not make use of the broadcast channel.*

From the above lemmas, it is clear that assuming one-way functions, we can stick to the following conventions:

- While proving lower bounds, we show that there is no n-party t-fair coin tossing protocol (as defined in Section 2.3) in an (n, β)-unreactive world for certain values of n, t, β. From Lemma 3, this shows that there is no n-party protocol that realizes the functionality $\mathcal{F}_{\mathsf{coin}}$ (see Figure 2) with fairness in the presence of t corruptions in a hybrid model where parties have access to unreactive functionalities of cardinality upper bounded by β (that is, an (n, β)-unreactive world without broadcast). In particular, this shows that unreactive functionalities of cardinality β are incomplete for n-party fair MPC in the presence of t corruptions.
- While proving upper bounds, or designing protocols to realize any arbitrary functionality $\mathcal{F}$ in the presence of t corruptions, we will construct protocols that achieve security with fairness against t corruptions in our (n, β)-unreactive world for certain values of n, t, β. From Lemma 4, this shows that there exist protocols that realize $\mathcal{F}$ with fairness in the presence of t corruptions in our (n, β)-unreactive world without broadcast. In particular, this shows that unreactive functionalities of cardinality β are complete for n-party fair MPC in the presence of t corruptions.

2.8 Broadcast

Broadcast is defined as in Figure 3. We recall that the ideal functionality for broadcast, namely $\mathcal{F}_{\mathsf{bc}}$, can be securely computed with guaranteed output delivery in the presence of t-threshold adversaries if and only if $0 \leq t < n/3$ [PSL80, LSP82]. Furthermore, $\mathcal{F}_{\mathsf{bc}}$ can be securely computed with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ [FGH$^+$02]. Furthermore, these results hold irrespective of the model we are working in as long as we do not have explicit access to $\mathcal{F}_{\mathsf{bc}}$.

Preliminaries: $x_1, x_2 \in \{0,1\}^*$; f_1, f_2 are 2-input, 2-output functions; ϕ_1, ϕ_2 are boolean predicates. The functionality proceeds as follows:

- **Input phase.** Upon receiving inputs $(x_1, f = (f_1, f_2, \phi_1, \phi_2))$ from P_1 and (x_2, f') from P_2, check if $f = f'$. If not, abort. Else, compute $f_1(x_1, x_2)$. If $f_1(x_1, x_2) = \perp$[a], abort. Else, send $f_1(x_1, x_2)$ to both parties, and go to next phase.
- **Trigger phase.** Upon receiving input w from party P_i, check if $\phi_i(w) = 1$. If yes, then send $(w, f_2(x_1, x_2, w))$ to both P_1 and P_2.

[a] We crucially require that $\perp$ is a special symbol different from the empty string. We use $\perp$ as a means of signalling that the input phase of $\mathcal{F}_{\mathsf{SyX}}$ did not complete successfully. We will however allow parties to attempt to invoke the input phase of the functionality at a later time. However, as we proceed, we will also have our functionality be clock-aware and thus only accept invocations to the input phase until a certain point in time. After the input phase times out, the functionality is rendered completely unusable. Similarly, if the input phase has been completed successfully, a clock-oblivious version of the functionality can be triggered at any point in time as long as a valid witness is provided, no matter the number of failed attempts. The clock-aware version of the functionality, however, will only accept invocations of the trigger phase until a certain point in time. After the trigger phase times out, the functionality is rendered completely unusable.

Figure 4: The ideal functionality $\mathcal{F}_{\mathsf{SyX}}$.

2.9 Synchronizable Exchange

Synchronizable exchange is defined as in Figure 4. In order to guarantee termination, we will need our ideal functionality to be "clock-aware". In this work, we stick to the formalism outlined in [PST17]. We recall that in this model, we assume that every party and every invocation of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ has access to a variable r that reflects the current round number. More generally, every function and predicate that is part of the specification of $\mathcal{F}_{\mathsf{SyX}}$ may also take r as an input. Finally, the functionality may also time out after a pre-programmed amount of time. We describe this clock-aware functionality in Figure 5. It is known that $\mathcal{F}_{\mathsf{SyX}}$ is complete for fair secure multiparty computation. We state this result formally below.

Lemma 5. *[KRS20] Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way functions, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \le t < n$ in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model.*

Lemma 6. *[KRS20] Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way permutations, there exists a protocol π in the programmable random oracle model which securely preprocesses for and computes an arbitrary (polynomial) number of instances of $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \le t < n$ in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model.*

3 Bypass Cleve's Lower Bound in Unreactive Worlds

As evidenced by Cleve's lower bound in [Cle86] for fair coin tossing in the presence of a dishonest majority, MPC with fairness is impossible in the presence of a dishonest majority if parties are

Preliminaries: $x_1, x_2 \in \{0,1\}^*$; f_1, f_2 are 2-output functions; ϕ_1, ϕ_2 are boolean predicates; r denotes the current round number; $\mathsf{INPUT\_TIMEOUT} < \mathsf{TRIGGER\_TIMEOUT}$ are round numbers representing time outs. The functionality proceeds as follows:

- **Load phase.** If $r > \mathsf{INPUT\_TIMEOUT}$, abort. Otherwise, upon receiving inputs of the form $(x_1, f = (f_1, f_2, \phi_1, \phi_2))$ from P_1 and (x_2, f') from P_2, check if $f = f'$. If not, abort. Else, compute $f_1(x_1, x_2, r)$. If $f_1(x_1, x_2, r) = \perp$, abort. Else, send $f_1^i(x_1, x_2, r)$ to P_i for $i \in \{1, 2\}$, and go to next phase.
- **Trigger phase.** If $r > \mathsf{TRIGGER\_TIMEOUT}$, abort. Otherwise, upon receiving input w from party P_i, check if $\phi_i(w, r) = 1$. If yes, then send $(w, f_2^j(x_1, x_2, w, r))$ to both parties P_j for $j \in \{1, 2\}$.

Figure 5: The clock-aware ideal functionality $\mathcal{F}_{\mathsf{SyX}}$.

only equipped with communication channels[15], even when the adversary is only allowed to corrupt $\lceil \frac{n}{2} \rceil$ parties. In this section, we present two simple and elegent MPC protocols with fairness in the presence of a dishonest majority by leveraging unreactive functionalities with of cardinality at least 2, which lets us bypass Cleve's lower bound. The existence of these protocols motivates the exploration of lower bounds in unreactive worlds, which is the main focus of this work. Meanwhile, it also shows that Cleve's technique in its vanilla form (and also, many similar techniques [IKK+11, HIK+19] based on [Cle86] from past works) is no longer sufficient to tackle the problem of proving lower bounds in the unreactive world. In general, consider n parties out of which t may be malicious ($\frac{n}{2} \le t < n$) in the (n, β)-unreactive world. Recall that β denotes the cardinality of the unreactive functionalities that are provided.

The case of $\beta > t$. Unsurprisingly, if parties have access to an unreactive functionality of cardinality $\beta > t$, there is a very simple and elegant protocol for fair MPC. Let $\mathbb{P} = \{P_1, \ldots, P_n\}$. The MPC protocol proceeds as follows.

1. All parties perform an unfair MPC that computes the result and β-out-of-β shares it to the parties $\{P_1, \ldots, P_\beta\}$.
2. $P_1, \ldots, P_\beta$ perform a β-wise exchange using the available β-wise functionality to reconstruct the result.
3. Each party P_i in $\{P_1, \ldots, P_\beta\}$ broadcasts the result if it obtained it in the previous step.

To see that this protocol achieves fairness, note that an adversary corrupting only t parties cannot learn the output in step 1 since it only has $t < \beta$ shares. To get the result, the adversary needs to let step 2 execute correctly. However, if step 2 executes correctly, all honest parties get the result as well. Some standard authentication techniques need to be used to ensure that the parties submit correct shares, etc. For more on such techniques, we refer the reader to [KRS20].

[15] More precisely, as long as the channels are *one-directional*, such as *OT channels*, Cleve's lower bound holds.

The case of $t = \frac{n}{2}$. The case of $t = \frac{n}{2}$ is rather interesting, and not representative of the case of $t \geq \frac{n}{2}$. In fact, 2-wise unreactive functionalities, in particular, 2-wise exchange suffices for fair MPC. Let $\mathbb{P} = \{P_1, \dots, P_n\}$. The MPC protocol proceeds as follows.

1. All parties perform an unfair MPC that computes the result and $(\frac{n}{2}+1)$-out-of-n shares it to all the parties.
2. For each $i, j \in [n]$ where $i < j$, parties P_i and P_j perform a 2-wise exchange for their shares.
3. For each $k \in [n]$, if a party P_k receives at least $\frac{n}{2}$ shares in the previous step, it broadcast any $(\frac{n}{2}+1)$ shares it has (including its own).
4. For each $k \in [n]$, if a party P_k has at least $\frac{n}{2}+1$ shares, it recovers the result.

To see that this protocol achieves fairness, an adversary corrupting only $\frac{n}{2}$ parties cannot learn the output in step 1 since it only has $\frac{n}{2}$ shares. To learn the output, the adversary needs to at least exchange one share with some honest party in step 2. Note that this honest party will get at least $\frac{n}{2}$ (1 from the adversary, $\frac{n}{2}-1$ from other honest parties) in step 2 so it already has enough shares to recover the result, and will broadcast the shares to all honest parties in step 3. Some standard authentication techniques need to be used to ensure that the parties submit correct shares, etc. For more on such techniques, we refer the reader to [KRS20].

4 Alice and Bob: Same World, Different Proofs

In Section 3, we showed that for any $\frac{n}{2} \leq t < n$, there exists an n-party MPC protocol with fairness in the (n, β)-unreactive world as long as $\beta > t$. Interestingly, we showed that there exists an n-party MPC protocol with fairness in the presence of $\frac{n}{2}$ corruptions in the $(n, 2)$-unreactive world. We remind the reader that [KRS20] presents an n-party MPC protocol with fairness in the presence of $n-1$ corruptions assuming the existence of 2-wise *reactive* functionalities, namely, in an $(n, 2)$-*"reactive" world*. A natural attempt is to construct an n-party MPC protocol with fairness in the presence of $n-1$ corruptions in the $(n, 2)$-unreactive world. Unfortunately, it turns out that for any $\frac{n}{2} < t < n$, there is no n-party MPC protocol with fairness in the (n, t)-unreactive world. This shows that our results are tight with our matching upper bounds and lower bounds. We will present these lower bounds by showing that there exists no n-party $(n-1)$-fair coin tossing protocol (see Definition 3) in the $(n, n-1)$-unreactive world in Section 5 and later generalize this lower bound to threshold adversaries in Section 6 for all $\frac{n}{2} < t < n$. As a warm-up, we show that there is no 2-party 1-fair coin tossing protocol in the $(2, 1)$-unreactive world. This is the most unsurprising case since the $(2, 1)$-unreactive world is identical to the two-party model used by [Cle86].

In the $(2, 1)$-unreactive world, 1-wise functionalities are local computations and the broadcast channel can be viewed as a communication channel between the two parties. Thus, it's identical to the two-party model used by [Cle86]. We emphasize that this is the *only* unreactive world that is not stronger than Cleve's. Our proof technique in Section 5 in the case of two parties can be viewed as a different take on the result of [Cle86]. Crucially, this allows us to generalize the lower bound of the $(2, 1)$-unreactive world to other unreactive worlds. We remind the reader that applying [Cle86]'s technique is not sufficient for unreactive worlds in general (and in fact, it only works for the $(2, 1)$-unreactive world) since we can easily bypass [Cle86]'s impossibility in some unreactive worlds as shown in Section 3. In this section, we briefly review the proof of [Cle86]'s lower bound in the case of two parties and show how our proof of the lower bound handles it differently. Looking ahead, our proof captures "fairness" in a more natural way by introducing a notion we call *predictability*.

A two-party coin tossing protocol has two parties, Alice and Bob, who share a communication channel. A round in the protocol corresponds to each party sending a message to the other. For simplicity, the protocol is assumed to be serialized such that Alice sends the first message and Bob sends the second message. Specifically, the protocol is captured as (1) Alice and Bob generate local randomness r_A and r_B; (2) for the next R rounds, Alice sends message $m_{B,i}$ followed by Bob's message $m_{A,i}$; and (3) Alice outputs $A(r_A, m_{A,1}, \ldots, m_{A,R})$ and Bob outputs $B(r_B, m_{B,1}, \ldots, m_{B,R})$. We assume that the protocol satisfies perfect agreement. That is, $A(r_A, \ldots, m_{A,R}) = B(r_B, \ldots, m_{B,R}) = res$.

4.1 Recalling the Lower Bound of [Cle86]

We recall the two-party lower bound of [Cle86]. Given any R-round fair coin tossing protocol, [Cle86] constructs $4R + 1$ adversarial strategies. One of these is an adversarial strategy where a malicious Alice quits[16] right at the beginning of the protocol. The remaining $4R$ adversarial strategies are split as $2R$ strategies where a malicious Alice quits at some point in the protocol and $2R$ strategies where a malicious Bob quits at some point in the protocol. The strategies are of the following form: "Execute until round $j - 1$ for $j \in [R]$, assume the other party quits and compute the final output $\widetilde{res}$. If $\widetilde{res} = b$ for $b \in \{0, 1\}$, quit now, or quit in the next round after sending a message in this round." For example, an adversary Alice will execute until getting message $m_{A,i}$ ($i \in [R]$) from Bob and decide to either (1) quit the protocol at this round if $A(r_A, \ldots, m_{A,i}, 0, \ldots, 0) = 0$; or (2) quit the protocol at the next round after sending $m_{B,i+1}$. The net bias of all these adversarial strategies turns out to be the non-negligible chance (over $\frac{1}{2}$) that the parties have in agreeing on the final output at the end of the protocol.

The intuition behind this proof can be seen as a sort of hybrid argument. Prior to any communication between Alice and Bob, they have no chance (over $\frac{1}{2}$) of agreeing on the final output, while at the end of the protocol, they have a non-negligible chance (over $\frac{1}{2}$) of agreeing on the final output. Thus, there is some message in the protocol after which their chance of agreeing on the final output must have changed non-negligibly. The party that sent said message would then have known with non-negligible advantage (over $\frac{1}{2}$) what the final output of the protocol would be, and hence can bias the output.

4.2 A Different Proof of [Cle86]: Predictability

In this section, we show how our proof works for the $(2, 1)$-unreactive world, which is identical to [Cle86]'s two-party model. Recall that the notion of fairness requires that the adversary cannot learn the output of the protocol without the honest parties learning it too. Naturally, this means that in an *unfair* coin tossing protocol, the adversary should be able to "predict" the output of the coin res at some point while the honest party cannot. Note that res, the honest output of the coin toss, should become "predictable" at the end of the protocol. Consider the following probabilities:

[16] Recall that if a party quits at some point, from that point onwards, all the messages that the party sends are zeros.

$$\begin{aligned}
\mathsf{Pred}_{A,0} &\triangleq \Pr[A(r_A, 0, \ldots) = res] \\
\mathsf{Pred}_{B,0} &\triangleq \Pr[B(r_B, 0, \ldots) = res] \\
\mathsf{Pred}_{A,1} &\triangleq \Pr[A(r_A, m_{A,1}, 0, \ldots) = res] \\
\mathsf{Pred}_{B,1} &\triangleq \Pr[B(r_B, m_{B,1}, 0, \ldots) = res] \\
&\ldots \\
\mathsf{Pred}_{A,R} &\triangleq \Pr[A(r_A, m_{A,1}, \ldots, m_{A,R}) = res] \\
\mathsf{Pred}_{B,R} &\triangleq \Pr[B(r_B, m_{B,1}, \ldots, m_{B,R}) = res]
\end{aligned}$$

We denote the *predictability* of Alice (resp. Bob) with i messages as $\mathsf{Pred}_{A,i}$ (resp. $\mathsf{Pred}_{B,i}$) where $\mathsf{Pred}_{A,i}$ (resp. $\mathsf{Pred}_{B,i}$) are defined as above. In other words, $\mathsf{Pred}_{A,i}$ can be viewed as Alice's ability to use the partial information she received in the first i rounds to figure out the output. Specifically, Alice uses her partial information and imagines Bob quits at that point. We further name the random variable associated with $\mathsf{Pred}_{A,i}$ (resp. $\mathsf{Pred}_{B,i}$) a *predictor* $\Pi_{A,i}$ (resp. $\Pi_{B,i}$). For example, $\Pi_{A,0} = A(r_A, 0, \ldots)$.

From definition of res, $\mathsf{Pred}_{A,R} = \mathsf{Pred}_{B,R} = 1$, capturing that the output of the protocol must be predictable in the end. We now argue that a fair coin tossing protocol must satisfy the following: the predictability of Alice and Bob at the beginning of the protocol should be statistically close to $\frac{1}{2}$. That is, $|\mathsf{Pred}_{A,0} - \frac{1}{2}| \leq \delta_1(\lambda), |\mathsf{Pred}_{B,0} - \frac{1}{2}| \leq \delta_2(\lambda)$ for some negligible functions $\delta_1(\lambda)$ and $\delta_2(\lambda)$.

Consider the following two adversarial strategies for Alice, $\mathcal{A}_b$ where $b \in \{0, 1\}$: Alice generates r_A and invokes the predictor $\Pi_{A,0}$; if the value returned by the predictor is b, Alice quits, otherwise plays honestly. If the protocol achieves a fair coin, $\mathcal{A}_b$'s bias on Bob's output must be negligible. That is, we have the following two negligible terms.

$$\left|\Pr[\Pi_{A,0} = 0 \wedge \Pi_{B,0} = 0] + \Pr[\Pi_{A,0} = 1 \wedge res = 0] - \frac{1}{2}\right|$$
$$\left|\Pr[\Pi_{A,0} = 1 \wedge \Pi_{B,0} = 1] + \Pr[\Pi_{A,0} = 0 \wedge res = 1] - \frac{1}{2}\right|$$

Therefore, $|\Pr[\Pi_{A,0} = \Pi_{B,0}] - \Pr[\Pi_{A,0} = res]| \leq \delta(\lambda)$ for some negligible function $\delta(\lambda)$. Note that $|\Pr[\Pi_{A,0} = 0] - \frac{1}{2}|$ must be negligible. This can be seen by considering an adversarial strategy for Bob where Bob unconditionally quits at the beginning of the protocol. Similarly, $|\Pr[\Pi_{B,0} = 0] - \frac{1}{2}|$ must also be negligible. Furthermore, since $\Pi_{B,0}$ and $\Pi_{A,0}$ are independent, $|\Pr[\Pi_{A,0} = \Pi_{B,0}] - \frac{1}{2}|$ must be negligible, which implies that $|\frac{1}{2} - \Pr[\Pi_{A,0} = res]|$ is negligible. That is, $|\mathsf{Pred}_{A,0} - \frac{1}{2}| \leq \delta_1(\lambda)$ for some negligible function $\delta_1(\lambda)$. Similarly, $|\mathsf{Pred}_{B,0} - \frac{1}{2}| \leq \delta_2(\lambda)$ for some negligible function $\delta_2(\lambda)$.

We then consider how these predictors are related to each other. Consider the point in the protocol right after the first message $m_{B,1}$ is delivered from Alice to Bob. That is, Bob holds $r_B, m_{B,1}$ while Alice only holds r_A. Crucially, Bob can currently launch an attack to conditionally let Alice output $\Pi_{A,0}$. Similarly, Alice can conditionally let Bob output $\Pi_{B,1}$. Looking ahead, we will show that if $|\mathsf{Pred}_{B,1} - \mathsf{Pred}_{A,0}|$ is non-negligible, there is an adversarial strategy for either Alice or Bob that can bias the other (honest) party's output by this non-negligibly. We now argue the existence of a pair of predictabilities that differ non-negligibly. We say that such a pair induces a non-negligible *gap*. Consider the following triangle inequality:

$$\sum_{i=1}^{R} |\mathsf{Pred}_{B,i} - \mathsf{Pred}_{A,i-1}| + |\mathsf{Pred}_{A,i} - \mathsf{Pred}_{B,i}| \geq |\mathsf{Pred}_{A,R} - \mathsf{Pred}_{A,0}| \tag{1}$$

Note that $|\mathsf{Pred}_{A,R} - \mathsf{Pred}_{A,0}| \geq \frac{1}{2} - \delta(\lambda)$ for some negligible function $\delta(\lambda)$. A straightforward averaging argument indicates that at least one term of the left hand side should be statistically close to $\frac{1}{2R}$. Crucially, every single term on the left hand side reflects some middle point of the entire execution. For example, $|\mathsf{Pred}_{A,i} - \mathsf{Pred}_{B,i}|$ is where both parties finish i rounds.

Without loss of generality, assume that $|\mathsf{Pred}_{B,1} - \mathsf{Pred}_{A,0}| \geq \frac{1}{2R} - \delta(\lambda)$ for some negligible function $\delta(\lambda)$. Consider the following two adversarial strategies for Bob, $\mathcal{A}_{B,b}$ where $b \in \{0,1\}$: (1) Bob gets $m_{B,1}$ from Alice; (2) invokes the predictor $\Pi_{B,1}$; and (3) quits if the result is b, and plays honestly otherwise. The biases induced on Alice's output bias_b^A by $\mathcal{A}_{B,b}$ will be:

$$\mathsf{bias}_b^A = \left|\Pr[\Pi_{B,1} = b \wedge \Pi_{A,0} = b] + \Pr[\Pi_{B,1} = 1 - b \wedge res = b] - \frac{1}{2}\right|$$

Similarly, consider the following two adversarial strategies for Alice, $\mathcal{A}_{A,b}$ where $b \in \{0,1\}$: (1) Alice sends $m_{B,1}$ to Bob; (2) invokes the predictor $A(r_A, 0, \ldots)$; and (3) quits if the result is b, and plays honestly otherwise. The biases induced on Bob's output bias_b^B by $\mathcal{A}_{A,b}$ will be:

$$\mathsf{bias}_b^B = \left|\Pr[\Pi_{A,0} = b \wedge \Pi_{B,1} = b] + \Pr[\Pi_{A,0} = 1 - b \wedge res = b] - \frac{1}{2}\right|$$

Consider the sum of these four biases:

$$\begin{aligned}
&\mathsf{bias}_0^A + \mathsf{bias}_1^A + \mathsf{bias}_0^B + \mathsf{bias}_1^B \\
&= \left|\Pr[\Pi_{B,1} = 0 \wedge \Pi_{A,0} = 0] + \Pr[\Pi_{B,1} = 1 \wedge res = 0] - \frac{1}{2}\right| \\
&\qquad + \left|\Pr[\Pi_{B,1} = 1 \wedge \Pi_{A,0} = 1] + \Pr[\Pi_{B,1} = 0 \wedge res = 1] - \frac{1}{2}\right| \\
&\qquad + \left|\Pr[\Pi_{A,0} = 0 \wedge \Pi_{B,1} = 0] + \Pr[\Pi_{A,0} = 1 \wedge res = 0] - \frac{1}{2}\right| \\
&\qquad + \left|\Pr[\Pi_{A,0} = 1 \wedge \Pi_{B,1} = 1] + \Pr[\Pi_{A,0} = 0 \wedge res = 1] - \frac{1}{2}\right| \\
&\geq |\Pr[\Pi_{B,1} = res] - \Pr[\Pi_{B,1} = \Pi_{A,0}]| + |\Pr[\Pi_{A,0} = \Pi_{B,1}] - \Pr[\Pi_{A,0} = res]| \\
&\geq |\Pr[\Pi_{B,1} = res] - \Pr[\Pi_{A,0} = res]| \\
&= |\mathsf{Pred}_{B,1} - \mathsf{Pred}_{A,0}| \geq \frac{1}{2R} - \delta(\lambda)
\end{aligned}$$

As a result, at least 1 out of these 4 adversarial strategies can induce a $\Omega(\frac{1}{R})$ bias on the output of the corresponding honest party. We emphasize that the two adversarial strategies we construct for Alice do not use her *latest* predictor. Namely, even though Alice can decide to quit and let Bob output $\Pi_{B,1}$ after receiving $m_{A,1}$, she does not utilize the predictor $\Pi_{A,1}$.

No matter which term on the left-hand side of Equation (1) induces a non-negligible gap[17], we can mimic the above to construct 4 adversarial strategies where at least 1 of them will induce a non-negligible bias.

We end this section by summarizing how our adversary is constructed and why it works. Our attack in the case of two parties essentially relies on finding a non-negligible gap between the predictabilities of two parties. But we require more. Consider the pair of predictors associated with the predictabilities that differ non-negigibly. Our proof technique relies on the fact that there exists two adversarial strategies, each using one of the predictors and forcing the honest party to output the result of the other predictor. Apart from being intuitive, this is crucial for our technique since it allows us to construct a telescoping sum such as in Equation (1) *a la* [Cle86]. Doing so reproduces the non-negligible gap which we can then utilize to argue the existence of adversarial strategies that induce non-negligible bias. For example, the pair of predictors $\Pi_{B,2}$ and $\Pi_{A,0}$ is suitable for our technique. This is because an adversarial strategy that uses $\Pi_{B,2}$ cannot force Alice to output the result of $\Pi_{A,0}$ as at this point in the protocol, Alice already has her first message $m_{A,1}$ from Bob. Looking ahead, the ability to find the above suitable gap is the core methodology we use to generalize this proof strategy to the case of n parties in the presence of $n-1$ corruptions in the $(n, n-1)$-unreactive world (see Section 5) and threshold corruptions (see Section 6).

4.3 Comparison

Predict-and-quit adversaries. The adversarial strategies we have designed are what we will call *predict-and-quit* strategies as they all execute the protocol honestly until a certain point, invoke some predictor and then decide whether to continue the protocol honestly to completion or quit right then. Note that the adversary uses the same computational power as that of an honest party since what it does is specified entirely by the protocol description. We emphasize that predict-and-quit adversaries are not the same as *fail-stop* adversaries [CI93] discussed in the literature. A fail-stop adversary is a *computationally unbounded* adversary that follows the protocol honestly and quits at some point. While a fail-stop adversary can possibly quit in two (or more) different rounds, a predict-and-quit adversary only quits in one specific round, if it does. Note that a predict-and-quit adversary is also a fail-stop adversary, but not other way around. Looking ahead, all adversaries we construct in this work are predict-and-quit adversaries.

Comparing with Cleve [Cle86]. Indeed, [Cle86]'s adversaries may seem eerily similar to ours, the only difference being that Cleve's adversaries quit in a certain round or the next.[18] However, we argue that the underlying technique is totally different and the difference captures why [Cle86]'s lower bounds do not work in other unreactive worlds. The main idea in [Cle86] to generalize its two-party lower-bound to the case of multiple parties is by arguing that parties can be split into two sets. These two sets can be viewed as virtual Alice and virtual Bob. Thus, [Cle86] can generalize

[17] Note $|\mathsf{Pred}_{A,R} - \mathsf{Pred}_{B,R}| = 0$, so the gap won't be in this term.

[18] An interesting note: Cleve's adversaries are fail-stop but not predict-and-quit.

its two-party technique to the multi-party setting naturally by considering the relationship of two predictors of these two *disjoint* sets. This immediately fails in other unreactive worlds where unreactive functionalities may render it impossible to consider such disjoint sets. As we will present, our lower bounds in the multi-party setting rely on considering the relationship of two predictors of two *overlapping* sets. For example, we may construct and analyze a pair of adversarial strategies one for $\{A, B\}$ attacking $\{C\}$, and another for $\{B, C\}$ attacking $\{A\}$. Our technique, particularly the notion of predictors and predictabilities, is what allows us to first argue the existence of a non-negligible gap and then construct the adversarial strategies accordingly.

5 All-but-one Corruptions in Unreactive Worlds

In this section, we show how we can extend our impossibility proof technique from Section 4.2 to the multi-party unreactive worlds in the presence of all-but-one corruptions. We will show that there exists no n-party $(n-1)$-fair coin tossing protocol in the $(n, n-1)$-unreactive world. That is, in the presence of $n-1$ corruptions, $(n-1)$-wise unreactive functionalities are insufficient for fairness. Recall that, without loss of generality, an n-party coin tossing protocol in the $(n, n-1)$-unreactive world can be captured by the $(n, R, n-1)$-unreactive syntax.

An n-party coin tossing protocol following the $(n, R, n-1)$-unreactive syntax is an R-round protocol, where each party will get $2R(n-1)$ messages from functionalities and the broadcast channel. Specifically, in each round, the functionality without P_1 is enabled first, the functionality without P_2 is enabled second, ..., the functionality without P_n is enabled nth and each party (from P_1 to P_n) becomes broadcaster in order. We abstract the description of the protocol as n procedures $\{\Pi_1, \ldots, \Pi_n\}$. Each party i will (1) generate randomness r_i; (2) receive $2R(n-1)$ messages $\{m_{i,1}, \ldots, m_{i,2R(n-1)}\}$ in sequence; and (3) output $\Pi_i(r_i, m_{i,1}, \ldots, m_{i,2R(n-1)})$. Note that some messages are delivered synchronously. For example, after the first functionality connected to $\mathbb{P} \setminus \{P_1\}$ is executed, all parties except P_1 will get their first message simultaneously.

We assume that the protocol achieves perfect agreement. That is, $\Pi_i(r_i, m_{i,1}, \ldots, m_{i,2R(n-1)}) = \Pi_j(r_j, m_{j,1}, \ldots, m_{j,2R(n-1)}) = res$ for any i and j.

5.1 Generalizing Predictors and Predictabilities

Recall that in two-party setting, the predictor with i messages of Alice is defined by the random variable $A(r_a, m_{A,1}, \ldots, m_{A,i}, 0, \ldots)$. That is, it calculates Alice's output when Alice has received her first i messages correctly, and the rest messages from then are 0s caused by Bob having quit at that point. Furthermore, the predictability with i messages is defined by the probability that the output of the corresponding predictor is equal to the honest output res. We can extend predictors and predictabilities as follows.

Definition 6 (Predictor/Predictability, n-party, $(n-1)$-corruptions)**.** *For an n-party coin tossing protocol among parties $\mathbb{P} \triangleq \{P_1, \ldots, P_n\}$, the predictor of party P_i with j messages is the output of an honest party P_i, where P_i is executed with malicious $\mathbb{P} \setminus \{P_i\}$ such that the adversary will follow the protocol honestly to allow P_i to obtain its first j messages correctly and then quits. We denote this predictor by $\Pi_{i,j}$. The predictability of party P_i with j messages is defined as $\mathsf{Pred}_{i,j} \triangleq \Pr[\Pi_{i,j} = res]$ where res is the output of the honest executed protocol (assuming perfect agreement). The probabilities are taken over the randomness of all parties and hybrid functionalities.*

Remark 1. *Note that our definition of a predictor and predictability is* independent *of the model. Furthermore, we define predictors for each* individual *party. This choice is guided by the fact that we consider all-but-one corruptions.*

Consider the predictors/predictabilities for a coin tossing protocol following the $(n, R, n-1)$-unreactive syntax. For example, $\Pi_{1,1}$ is the output of an honest P_1 executed with malicious $\mathbb{P} \setminus \{P_1\}$ such that the adversary (1) generates correct randomness; (2) participates in two unreactive functionalities correctly where they are connected with $\mathbb{P} \setminus \{P_1\}$ and $\mathbb{P} \setminus \{P_2\}$, which will send the first message to P_1 correctly; and (3) quits by sending zeros to all other functionalities. Note that each party will get $2R(n-1)$ messages in total in an honest execution of this protocol. From the agreement (assuming perfect) requirement of coin tossing protocol, we know $\Pi_{i,2R(n-1)} = res$ for all $i \in [n]$. This implies that the predictability $\mathsf{Pred}_{i,2R(n-1)} = 1$. Similar to the two-party setting, if the n-party coin tossing protocol is $(n-1)$-*fair*, we now argue that the predictability of each party at the beginning of the protocol (i.e., after seeing 0 messages) should be statistically close to $\frac{1}{2}$. For each $i \in [n]$, we call $\Pi_{i,0}$ the initial predictor for P_i and the $\Pi_{i,2R(n-1)}$ the final predictor for P_i.

Lemma 7. *Consider an n-party coin tossing protocol among parties $\mathbb{P} \triangleq \{P_1, \ldots, P_n\}$ and the associated predictors/predictabilities (Cf. Definition 6). If the protocol is* $(n-1)$-fair *(Cf. Definition 4), then for all $i \in [n]$,*

$$\left|\mathsf{Pred}_{i,0} - \frac{1}{2}\right| \leq \delta^{(i)}(\lambda)$$

for some negligible function $\delta^{(i)}(\lambda)$.

Proof. Without loss of generality, consider $i = 1$. Since the protocol is fair, the output of P_1 when executing with malicious $\mathbb{P} \setminus \{P_1\}$, where the adversary simply quits at the beginning, should be a fair coin, that is,

$$\left|\Pr[\Pi_{1,0} = 0] - \frac{1}{2}\right| \leq \delta_1(\lambda)$$

for some negligible function $\delta_1(\lambda)$. Similarly, $|\Pr[\Pi_{2,0} = 0] - \frac{1}{2}| \leq \delta_2(\lambda)$ for some negligible function $\delta_2(\lambda)$. Consider the following two adversarial strategies $\mathcal{A}_b$ ($b \in \{0,1\}$) for $\mathbb{P} \setminus \{P_2\}$: $\mathcal{A}_b$ (1) generates the randomness of $\mathbb{P} \setminus \{P_2\}$ correctly; (2) invokes $\Pi_{1,0}$ (note that this random variable does not rely on P_2's randomness); and (3) quits if the result is b, and follows the protocol honestly otherwise. If the protocol is fair, the bias bias_b of P_2's output induced by $\mathcal{A}_b$ should be negligible. Formally,

$$\mathsf{bias}_0 = \left|\Pr[\Pi_{1,0} = 0 \wedge \Pi_{2,0} = 0] + \Pr[\Pi_{1,0} = 1 \wedge res = 0] - \frac{1}{2}\right| \leq \delta_3(\lambda)$$

$$\mathsf{bias}_1 = \left|\Pr[\Pi_{1,0} = 1 \wedge \Pi_{2,0} = 1] + \Pr[\Pi_{1,0} = 0 \wedge res = 1] - \frac{1}{2}\right| \leq \delta_4(\lambda)$$

where $\delta_3(\lambda), \delta_4(\lambda)$ are some negligible functions, which implies

$$|\Pr[\Pi_{1,0} = \Pi_{2,0}] - \Pr[\Pi_{1,0} = res]| \leq \delta_5(\lambda)$$

for some negligible function $\delta_5(\lambda)$. Note that $\Pi_{1,0}$ and $\Pi_{2,0}$ are independent, $|\Pr[\Pi_{1,0} = 0] - \frac{1}{2}| \leq \delta_1(\lambda)$ and $|\Pr[\Pi_{2,0} = 0] - \frac{1}{2}| \leq \delta_2(\lambda)$. Therefore,

$$\left|\Pr[\Pi_{1,0} = res] - \frac{1}{2}\right| \leq \delta_6(\lambda) \Leftrightarrow \left|\mathsf{Pred}_{i,0} - \frac{1}{2}\right| \leq \delta_6(\lambda)$$

for some negligible function $\delta_6(\lambda)$. A similar argument applies for $i \neq 1$. $\square$

We end this section by noting that if an n-party coin tossing protocol is $(n-1)$-fair, the predictability gap between $\Pi_{1,0}$ and $\Pi_{n,2R(n-1)}$ must be $\Omega(1)$. We next show how this gap implies an attackable gap as in the two-party setting.

5.2 Attackable Non-negligible Gaps

Recall how we constructed adversaries in the two-party setting. Specifically, we construct 4 predict-and-quit adversaries. These 4 adversaries rely on two underlying predictors – one predictor $\Pi_{A,i}$ of Alice and another predictor $\Pi_{B,j}$ of Bob. Crucially, they need to satisfy the following two requirements:

1. These two predictors are interchangeably attackable, and form an attackable pair. That is, when a malicious Alice holds enough information to calculate $\Pi_{A,i}$, she should still be able to let Bob output $\Pi_{B,j}$. A similar requirement holds for a malicious Bob.
2. The predictability of these two predictors has a gap, namely, $|\Pr[\Pi_{A,i} = res] - \Pr[\Pi_{B,j} = res]| \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\lambda)$.

Note that in the two-party setting where Alice and Bob send messages in sequence, when Bob has j messages from Alice, Alice should already have received $j-1$ messages. This means that a predict-and-quit Bob based on the predictor $\Pi_{B,j}$ can only conditionally let Alice output $\Pi_{A,k}$ where $k \geq j-1$. Similarly, when Alice has i messages from Bob, Bob should already have received i messages. That means that a predict-and-quit Alice based on the predictor $\Pi_{A,i}$ can only conditionally let Bob output $\Pi_{B,k}$ where $k \geq i$. Thus, in order to satisfy bullet point 1 above, we will have $j \geq i$ and $i \geq j-1$, that is, $j = i$ or $j = i+1$.

To show that there is an attackable predictor pair satisfying bullet point 2, we observe that the predictability gap between $\mathsf{Pred}_{A,0}$ and $\mathsf{Pred}_{B,R}$ must be $\Omega(1)$ in a two-party R-round *fair* coin tossing protocol. Imagine a graph where each vertex represents a predictor and two vertices share an edge if and only if they can form an attackable pair (Cf., bullet point 1). Crucially, there exists a predictor path of length $\mathcal{O}(R)$ in the graph from $\Pi_{A,0}$ to $\Pi_{B,R}$, namely $\Pi_{A,0} \to \Pi_{B,1} \to \Pi_{A,1} \to \cdots \to \Pi_{B,R}$. Therefore, by the triangle inequality, there exists at least one attackable predictor pair (i.e., an edge in the graph) on the path such that their predictability gap is $\Omega(\frac{1}{R})$.

We now transplant the above idea to the n-party setting in the $(n, n-1)$-unreactive world. In fact, using our new predictor/predictability notion, if an n-party coin tossing protocol (following the $(n, R, n-1)$-unreactive syntax) has two predictors such that (1) their predictabilities have a non-negligible gap; and (2) they can attack each other interchangeably, we can mimic the 4 adversaries we constructed in the two-party setting where at least 1 of them will induce a non-negligible bias. For example, assume there is a non-negligible gap between $\mathsf{Pred}_{1,0}$ and $\mathsf{Pred}_{2,1}$. We can construct two adversaries corrupting $\mathbb{P} \setminus \{P_2\}$ (resp. $\mathbb{P} \setminus \{P_1\}$) that based on the output of $\Pi_{1,0}$ (resp. $\Pi_{2,1}$) conditionally let P_2 (resp. P_1) output $\Pi_{2,1}$ (resp. $\Pi_{1,0}$). We emphasize the underlying reason why an adversary (corrupting $\mathbb{P} \setminus \{P_1\}$) can let P_1 output $\Pi_{1,0}$ based on $\Pi_{2,1}$: the adversary can get the first message of P_2 without letting P_1 get its first message since the first unreactive functionality is among $\mathbb{P} \setminus \{P_1\}$. A predictor may not always be able to "attack" another predictor. For example, $\Pi_{2,1}$ cannot "attack" $\Pi_{3,0}$. Formally, an attackable predictor pair is defined as follows.

Definition 7 (Attackable Pair)**.** *Consider an n-party coin tossing protocol and associated predictors/predictabilities (Cf. Definition 6).* Π_{i_1,j_1} *and* Π_{i_2,j_2} *form an attackable predictor pair if they satisfy the following properties:*

- $i_1 \neq i_2$.
- *When an adversary corrupting $\mathbb{P} \setminus \{P_{i_1}\}$ (resp. $\mathbb{P} \setminus \{P_{i_2}\}$) obtains sufficient information to calculate Π_{i_2,j_2} (resp. Π_{i_1,j_1}), it can still let P_{i_1} (resp. P_{i_2}) output Π_{i_1,j_1} (resp. Π_{i_2,j_2}) by quitting.*

Remark 2. *In any n-party coin tossing protocol, after the invocation of any hybrid functionality (unreactive or broadcast), the* latest *predictors of any two different parties form an attackable pair.*

In the previous section, we show that an n-party fair coin tossing protocol must have an $\Omega(\frac{1}{2})$ gap between $\mathsf{Pred}_{1,0}$ and $\mathsf{Pred}_{n,2R(n-1)}$. Again, imagine a graph where each node represents a predictor and two nodes share an edge if and only if they can form an attackable pair (Cf. Definition 7). We call this graph the *predictor graph*. If there is a (polynomial-length) path connecting $\Pi_{1,0}$ and $\Pi_{n,2R(n-1)}$ in the graph, we can argue the existence of an attackable pair whose predictabilities differ non-negligibly. Recall that the n parties are connected via $(n-1)$-wise unreactive functionalities and a broadcast channel. For all $k \in [R]$, round k proceeds as follows: $\mathcal{F}^{(k)}_{\mathbb{P}\setminus\{P_1\}}, \mathcal{F}^{(k)}_{\mathbb{P}\setminus\{P_2\}}, \ldots, \mathcal{F}^{(k)}_{\mathbb{P}\setminus\{P_n\}}$, P_1 broadcasts, P_2 broadcasts, …, P_n broadcasts. We have the following lemma.

Lemma 8. *Consider an n-party coin tossing protocol following the $(n, R, n-1)$-unreactive syntax and associated predictors/predictabilities (Cf. Definition 6).*

- *(***right after all unreactive functionalities/broadcast***) For any $1 \le p \le 2R$, $(\Pi_{1,p(n-1)}, \Pi_{n,p(n-1)})$ is an attackable pair.*
- *(***right after each unreactive functionality/broadcast***) For any $i \in [n-1]$, any $0 \le p < 2R$, $(\Pi_{i+1,p(n-1)+i}, \Pi_{i,p(n-1)+i-1})$ is an attackable pair.*

Proof. Consider an *honest* execution of the protocol. For any $1 \le p \le 2R$, there is a certain point where all parties (including P_1 and P_n) get exactly $p(n-1)$ messages. Namely, the execution is in round $\lfloor \frac{p+1}{2} \rfloor$ where either (1) the unreactive functionality connecting $\mathbb{P} \setminus \{P_n\}$ has just been executed; or (2) P_n has just completed its broadcast. Thus, an adversary corrupting $\mathbb{P} \setminus \{P_1\}$ (resp. $\mathbb{P} \setminus \{P_n\}$) can let P_1 (resp. P_n) output $\Pi_{1,p(n-1)}$ (resp. $\Pi_{n,p(n-1)}$) based on $\Pi_{n,p(n-1)}$ (resp. $\Pi_{1,p(n-1)}$). Thus, $(\Pi_{1,p(n-1)}, \Pi_{n,p(n-1)})$ is an attackable pair (Cf. Remark 2).

For any $i \in [n-1]$, any $0 \le p < 2R$, there is a certain point where P_{i+1} gets $p(n-1)+i$ messages but P_i only gets $p(n-1)+i-1$ messages. Namely, the execution is in round $\lfloor \frac{p+2}{2} \rfloor$ where either (1) the unreactive functionality connecting $\mathbb{P} \setminus \{P_i\}$ has just been executed; or (2) P_i has just completed its broadcast. Thus, $(\Pi_{i+1,p(n-1)+i}, \Pi_{i,p(n-1)+i-1})$ is an attackable pair (Cf. Remark 2). □

By Lemma 8, there exists a path from $\Pi_{1,0}$ to $\Pi_{n,2R(n-1)}$ in the predictor graph of length $\mathcal{O}(nR)$: $\Pi_{1,0} \to \Pi_{2,1} \to \Pi_{3,2} \to \cdots \to \Pi_{n,n-1} \to \Pi_{1,n-1} \to \Pi_{2,n} \to \cdots \to \Pi_{n,2R(n-1)}$. As a result, there exists at least one attackable predictor pair (i.e., an edge on the path) such that their predictability gap is $\Omega(\frac{1}{2nR})$. Without loss of generality, assume $|\mathsf{Pred}_{2,1} - \mathsf{Pred}_{1,0}|$ is $\Omega(\frac{1}{nR})$. By mimicking the 4 adversaries we constructed in the two-party setting, there exists at least 1 adversary corrupting $\mathbb{P} \setminus \{P_1\}$ (or $\mathbb{P} \setminus \{P_2\}$) that can bias the output of P_1 (or P_2) by $\Omega(\frac{1}{nR})$.

Theorem 1. *For any n-party coin tossing protocol in the $(n, n-1)$-unreactive world, following the $(n, R, n-1)$-unreactive syntax, there exists a predict-and-quit adversarial strategy corrupting $n-1$ parties that can bias the output of the honest party by $\Omega(\frac{1}{nR})$.*

We conclude this section by giving an intuitive reason why the above attackable predictor pairs should be considered. Note that the protocol execution is a sequence of invocations of hybrid functionalities ($(n-1)$-wise unreactive functionalities and the broadcast channel). Since broadcast can be viewed as a primitive delivering useful output to all parties except the broadcaster, each hybrid functionality delivers messages to $n-1$ parties. Consider these hybrid functionalities one by one. After the kth hybrid functionality is invoked, we consider the predictor pair formed by the *latest* predictors of (1) the party A_k who does not connect to this primitive; and (2) the party B_k who does not connect to the next primitive. By Remark 2, they form an attackable pair. Furthermore, all these attackable pairs will form a chain since B_k is precisely A_{k+1} and, importantly, the next primitive will not deliver any message to B_k/A_{k+1}[19], so the latest predictor of B_k/A_{k+1} stays the same. This chain will begin with some initial predictor[20] and end at some final predictor, and we know that the predictabilities corresponding to any intial predictor and any final predictor have a gap of $\Omega(1)$. This exactly reflects the path we have presented above.

5.3 Communication Channels v.s. Unreactive $\mathcal{F}$s v.s. Reactive $\mathcal{F}$s

[Cle86] shows that there is no n-party $\lceil \frac{n}{2} \rceil$-fair coin tossing protocol in a model where parties are connected via communication channels. In Section 3, we show that by leveraging $(n-1)$-wise unreactive functionalities, Cleve's impossibility result can be easily bypassed. In this section, we show that there is no n-party $(n-1)$-fair coin tossing protocol even using $(n-1)$-wise unreactive functionalities. However, our impossibility can also be bypassed by leveraging reactive functionalities as shown in [KRS20].

This might be counter-intuitive since our proof technique is general. Namely, one can apply our predictor/predictability framework to Cleve's model or even the one with reactive functionalities in [KRS20]. What causes the difference in results when we apply our proof techniques in different models? The difference lies in the attackable predictor pairs. Informally, there will be potentially fewer attackable predictor pairs in the reactive model. In the same vein, there will be potentially more attackable predictor pairs in [Cle86]'s model.

For the remainder of this section, consider a 4-party protocol between A, B, C, D.

Communication Channels v.s. Unreactive Functionalities. Consider a communication channel of A's sending messages to $\{B, C, D\}$ and an unreactive functionality connecting $\{B, C, D\}$. After both hybrid functionalities, B, C, D each gets one new message. Assume they are each the first hybrid functionality in two models. In the model where the communication channel is the first hybrid functionality, malicious (A, B) can leverage $\Pi_{B,1}$ to let C output $\Pi_{C,0}$ since by colluding with A, B can know his first message in a rushing manner. However, in the other model, malicious (A, B) can only leverage $\Pi_{B,1}$ after correctly enabling $\mathcal{F}^{(1)}_{B,C,D}$. Crucially, they cannot let C output $\Pi_{C,0}$ anymore.

[19] B_k/A_{k+1} is either not in the next unreactive functionality or it is the next broadcaster.

[20] Note that after the first unreactive functionality is enabled, the predictor of the party being "kicked-out" is still an initial predictor.

Unreactive Functionalities v.s. Reactive Functionalities. Consider two consecutive unreactive functionalities connecting $\{B, C, D\}$ and a 2-phase reactive functionality connecting $\{B, C, D\}$. Suppose the two consecutive unreactive functionalities aim to functionally simulate the 2-phase reactive functionality using secret sharing.[21] Assume they are each the first hybrid functionality(ies) in two models. In the model where the unreactive functionalities are the first hybrid functionalities, malicious (A, C, D) can invoke $\Pi_{C,1}$ to let B output $\Pi_{B,1}$ since they can enable $\mathcal{F}^{(1)}_{B,C,D}$[22] and quit. However, in the reactive model, if (A, C, D) quits after triggering the first phase of the reactive functionality to learn $\Pi_{C,1}$, B may output $\Pi_{B,2}$ by triggering the second phase of the reactive functionality. For the same reason, $\Pi_{B,2}$ may not form an attackable pair with $\Pi_{C,1}$. Thus, there may be no attackable predictor pair related to $\Pi_{C,1}$ in the reactive model.

6 Threshold Corruptions in Unreactive Worlds

In this section, we consider any $t > \frac{n}{2}$ in the (n, t)-unreactive world. As it turn out, t-wise unreactive functionalities are insufficient for n-party t-fair coin tossing for $t > \frac{n}{2}$. We will show this impossibility by further extending the notion of predictors/predictabilities to threshold adversaries. More importantly, we will show the existence of a non-negligible gap of predictability between an (extended) attackable predictor pair.

Note that all protocols in the (n, t)-unreactive world can be viewed as following the (n, R, t)-unreactive syntax. Recall that the main difference between the $(n, R, n-1)$-unreactive syntax and the (n, R, t)-unreactive syntax lies in the unreactive functionality phase in each round. That is, it will be a sequence of $\binom{n}{t}$ different t-wise unreactive functionalities connecting each subset of t parties. As a result, each party will receive $R(\binom{n-1}{t-1} + (n-1))$ in total. We do not specify the order of these t-wise unreactive functionalities in each round. In general, they can be arranged in any fixed order.

Throughout this section, consider an n-party coin tossing protocol following the (n, R, t)-unreactive syntax, where all the parties output res.

6.1 Generalizing Predictor and Predictability

When considering all-but-one corruptions, there will be a unique honest party. Thus, when an honest party is attacked by the other $n - 1$ parties by quitting, its following execution can be viewed as a local computation. However, when we consider t corruptions, even if t parties quit and start to forward zeros, there might still be information exchanged between honest parties. Therefore, we need to augment our predictors/predictabilities to support a set of honest parties.

Definition 8 (Predictor/Predictability, n-party, t-corruption)**.** *For an n-party coin tossing protocol among parties $\mathbb{P} \triangleq \{P_1, \dots, P_n\}$ in the presence of $t > \frac{n}{2}$ corruptions, the predictor of party P_i with j messages and an honest set H (where $H \subset \mathbb{P}$, $|H| = n - t$ and $P_i \in S$) is the output of the honest party P_i when H is executed honestly with malicious $\mathbb{P} \setminus H$ such that the adversary will follow the protocol honestly to allow P_i to obtain its first j messages correctly and the $(j+1)$th message incorrectly and then* immediately *quits. We denote this predictor by $\Pi_{i,j,H}$. The predictability of party P_i with j messages and an honest set H is defined as $\mathsf{Pred}_{i,j,H} \triangleq \Pr[\Pi_{i,j,H} = res]$ where res*

[21]In general, there is no such *secure* simulation. We only use this as an illustration.

[22]Simulate the first phase of the reactive functionality.

is the output of the honest executed protocol (assuming perfect agreement). The probabilities are taken over the randomness of all parties and hybrid functionalities.

Corollary 1. *We can extend the notion of predictors to consider a predictor of a set S. Formally, a predictor of a set S ($|S| = n - t$) with j messages is denoted by $\Pi_{P,j,S}$ where P is the party with smallest index in S. We may omit j when it's clear from context. The notion of predictabilities can be extended similarly.*

Definition 6 is a special case of Definition 8. For example, $\Pi_{i,j}$ in Definition 6 is the same as $\Pi_{i,j,\{P_i\}}$ in Definition 8.

Consider the predictors/predictabilities (Definition 8) of a protocol following the (n, R, t)-unreactive syntax. Obviously, for any i, any party set S where $P_i \in S$ and $|S| = n - t$, we have $\mathsf{Pred}_{i,R(\binom{n-1}{t-1}+(n-1)),S} = 1$. We now argue that if the protocol is t-*fair*, the initial predictabilities should be statistically close to $\frac{1}{2}$. For each i and compatible S, we call $\Pi_{i,0,S}$ the initial predictor and $\Pi_{i,R(\binom{n-1}{t-1}+(n-1)),S}$ the final predictor.

Lemma 9. *Consider an n-party coin tossing protocol in the presence of $t > \frac{n}{2}$ corruptions among parties $\mathbb{P} \triangleq \{P_1, \dots, P_n\}$ following the (n, R, t)-unreactive syntax and associated predictors/predictabilities (Cf. Definition 8). If the protocol is t-*fair*, then for all $i \in [n]$, for all $H \subseteq \mathbb{P}$ where $|H| = n - t$ and $P_i \in H$,*

$$\left|\mathsf{Pred}_{i,0,H} - \frac{1}{2}\right| \leq \delta^{(i,H)}(\lambda)$$

for some negligible function $\delta^{(i,H)}(\lambda)$.

Proof. Note that in the (n, R, t)-unreactive syntax, the unreactive functionalities are t-wise. Since $n-t < t$, for any unreactive functionality, there must be at least one party from $\mathbb{P}\backslash H$ is connecting to it. Since the unreactive functionalities are placed before the broadcasts in each round, an adversary corrupting $\mathbb{P} \setminus H$ can indeed let P_i obtain the first message incorrectly.

Let M be any subset of $\mathbb{P} \setminus H$ of size $n-t$. Let P_j be any party in M. Since the protocol is fair, the output of P_i when executing with malicious $\mathbb{P} \setminus H$ where the adversary aborts at the beginning should be a fair coin. This means that

$$\left|\Pr[\Pi_{i,0,H} = 0] - \frac{1}{2}\right| \leq \delta_1(\lambda)$$

for some negligible function $\delta_1(\lambda)$. Similarly, $|\Pr[\Pi_{j,0,M} = 0] - \frac{1}{2}| \leq \delta_2(\lambda)$ for some negligible function $\delta_2(\lambda)$. Consider the following two adversarial strategies, $\mathcal{A}_b$ ($b \in \{0, 1\}$), that corrupt $\mathbb{P} \setminus M$ and execute as follows: $\mathcal{A}_b$ (1) generates the randomness of $\mathbb{P} \setminus M$ correctly; (2) invokes $\Pi_{i,0,H}$[23]; and (3) quits if the result is b, and follows the protocol honestly otherwise. If the protocol is fair, the bias bias_b of P_j's output induced by $\mathcal{A}_b$ should be negligible. Formally,

$$\mathsf{bias}_0 = \left|\Pr[\Pi_{i,0,H} = 0 \wedge \Pi_{j,0,M} = 0] + \Pr[\Pi_{i,0,H} = 1 \wedge res = 0] - \frac{1}{2}\right| \leq \delta_3(\lambda)$$

$$\mathsf{bias}_1 = \left|\Pr[\Pi_{i,0,H} = 1 \wedge \Pi_{j,0,M} = 1] + \Pr[\Pi_{i,0,H} = 0 \wedge res = 1] - \frac{1}{2}\right| \leq \delta_4(\lambda)$$

[23] Note that the adversary controls all parties in H.

where $\delta_3(\lambda), \delta_4(\lambda)$ are some negligible functions. This implies that

$$|\Pr[\Pi_{i,0,H} = \Pi_{j,0,M}] - \Pr[\Pi_{i,0,H} = res]| \leq \delta_5(\lambda)$$

for some negligible function $\delta_5(\lambda)$. Note that $\Pi_{i,0,H}$ and $\Pi_{j,0,M}$ are independent, $|\Pr[\Pi_{i,0,H} = 0] - \frac{1}{2}| \leq \delta_1(\lambda)$ and $\| \Pr[\Pi_{j,0,M} = 0] - \frac{1}{2}| \leq \delta_2(\lambda)$. Therefore,

$$\left|\Pr[\Pi_{i,0,H} = res] - \frac{1}{2}\right| \leq \delta_6(\lambda) \Leftrightarrow \left|\mathsf{Pred}_{i,0,H} - \frac{1}{2}\right| \leq \delta_6(\lambda)$$

for some negligible function $\delta_6(\lambda)$. □

6.2 Attackable Non-negligible Gaps

From the above discussion, if an n-party coin tossing protocol following the (n, R, t)-unreactive syntax is t-fair, we know that any initial predictability is statiscally close to $\frac{1}{2}$ and any final predictability is 1. As in the case of $t = n - 1$, to construct a valid adversary, it suffices to find several (extended) attackable predictor pairs (Cf. Definition 9), which together forms a "chain" connecting an initial predictor and a final predictor.

Definition 9 (Attackable Pair, Extended). *Π_{i_1,j_1,H_1} and Π_{i_2,j_2,H_2} form a attackable pair if they satisfy the following properties:*

- *$H_1 \cap H_2 = \varnothing$.*
- *When an adversary corrupting $\mathbb{P} \backslash H_1$ (resp. $\mathbb{P} \backslash H_2$) obtains sufficient information to calculate Π_{i_2,j_2,H_2} (resp. Π_{i_1,j_1,H_1}), it can still let P_{i_1} (resp. P_{i_2}) output Π_{i_1,j_1,H_1} (resp. Π_{i_2,j_2,H_2}) by quitting.*

As in the case of $t = n - 1$, a natural way to find these pairs is by considering the point right after each hybrid functionality (i.e., t-wise unreactive functionalities and the broadcast channel) is executed in an honest execution. That is, after each hybrid functionality is executed, we append the predictor pair formed by the *latest* predictors of (1) the parties that do not participate in this hybrid functionality; and (2) the parties that do not participate in the next hybrid functionality. In the specific case of $t = n - 1$, this strategy directly induces a valid chain. However, there are following two major challenges in using this methodology when $t \neq n - 1$.

1. Since each unreactive functionality is connected to t parties, the sets of parties *not* participating in two consecutive unreactive functionalities may overlap. For example, consider two consecutive functionalities *not* connecting $\{A, B\}$ and $\{B, C\}$. Clearly, the predictor of $\{A, B\}$ and the predictor of $\{B, C\}$ cannot form an attackable pair. This is not an issue when $t = n - 1$.
2. The broadcast channel can be viewed as an $(n-1)$-wise functionality, namely, a party sending a message to all other parties. That is, only the broadcaster can be viewed as "not participating" in the broadcast channel. When $t \neq n - 1$, the broadcaster itself cannot form a predictor.

We present how we resolve these two challenges separately by rearranging unreactive functionalities in the syntax and considering the broadcast channel invocations in a batched manner. That is, we provide methods to construct a valid chain of attackable pairs if the syntax of the protocol only provided either unreactive functionalities or the broadcast channel. In the end, we explain how to solder these two types of chains together to get a valid chain for the protocol following the (n, R, t)-unreactive syntax.

6.2.1 Resolving Challenge 1: Rearrange Unreactive Functionalities.

Consider a protocol which only uses unreactive functionalities and no broadcast channel. That is, we only need to resolve challenge 1. Note that, as we mentioned, the syntax of the protocol can place the $\binom{n}{t}$ unreactive functionalities *in any order*. Thus, a straightforward solution is to consider whether we can place these functionalities in some order such that any two consecutive functionalities leave out disjoint sets of parties. We also need to satisfy this property for the first and the last functionality to allow for $R > 1$ rounds. This introduces a well-formed problem on graphs:

Consider an integer n, the set $N = [n]$ and an integer $\frac{n}{2} < t < n$. n and t induce an undirected graph (V, E) as follows: each vertex represents a subset of N of size $(n-t)$ (so $|V| = \binom{n}{n-t} = \binom{n}{t}$); two vertices share an edge if and only if two underlying sets are disjoint. Does (V, E) contain a Hamilton cycle*?*

Clearly, the introduced graph when $t = n - 1$ contains a Hamilton cycle, which reflects our attackable chain in Section 5.2. Incidentally, the above graph is called a *Kneser graph*, introduced by Lovász in [Lov78]. In graph theory, the Kneser graph $K(n, k)$ has as vertices all k-element subsets of an n-element ground set, and an edge between any two disjoint sets [MM+22]. That is, we are trying to find a Hamilton cycle in $K(n, n - t)$. Recently, Merino et al. [MM+22] proved that all Kneser graphs $K(n, k)$, where $n \geq 3$ and $0 < 2k < n$, admit a Hamilton cycle, except the well-known *Petersen graph* $K(5, 2)$. [24]

Therefore, for any $n \geq 3$ and $t > \frac{n}{2}$, except for $n = 5$ and $t = 3$, the Kneser graph $K(n, n - t)$ has a Hamilton cycle. We can order the t-wise functionalities in the order guided by the Hamilton cycle of the Kneser graph $K(n, n-t)$. In this way, after each functionality is executed, consider the predictor pair formed by the latest predictors of (1) the parties do not connect to this functionality; and (2) the parties do not connect to the next functionality. The predictors are of two disjoint sets, so they form an attackable pair. We emphasize that here we consider a Hamilton cycle rather than a *Hamilton path*. This is because with a cycle, this order can be repeated $R > 1$ times. Namely, *assume there is no broadcast*, the last functionality of some round is followed by the first functionality of the next round. We need to ensure this pair is attackable as well.

So far, we have constructed the attackable chain for any protocol that only uses unreactive functionalities, except for the special case of $n = 5, t = 3$, which we will resolve later.

6.2.2 Resolving Challenge 2: Batch Broadcast.

Consider a protocol which only uses a broadcast channel and no unreactive functionality. That is, we only need to resolve challenge 2. Consider a sequence of $n - t$ parties, who take turns to invoke the broadcast channel. As long as these $n - t$ parties are honest and the first broadcast happens correctly, these $n - t$ parties should all get $n - t - 1$ new correct messages. On the one hand, an adversary corrupting the other t parties can choose to only deliver either 0 or $n - t - 1$ new correct messages to each honest party for these broadcasts. On the other hand, an adversary corrupting

[24] This is a conjecture since the 1970s.

these $n-t$ parties can utilize the predictor based on all $n-t$ broadcast messages to decide whether to reveal all $n-t$ messages to the honest parties. Conceptually, these $n-t$ parties can be viewed as an entity where all broadcasts are batched into one.

Thus, we can view a sequence of broadcasts from P_1 to P_n as a sequence of $(n-t)$-batched broadcasts. That is, it can be viewed as a sequence of t-wise functionalities, where in each functionality, the t parties connecting to it will receive a batched broadcast from the other $(n-t)$ parties (i.e., the broadcasters). Specifically, the first batched broadcast will connect to $\mathbb{P} \setminus \{P_1, \ldots, P_{n-t}\}$, the second batched broadcast will connect to $\mathbb{P} \setminus \{P_{n-t+1}, \ldots, P_{2(n-t)}\}$ and so on. For the last batched broadcast, if $(n-t) \nmid n$, we add parties $P_1, \ldots, P_{(n-t)-(n \mod (n-t))}$ to the last batch as *dummy* broadcasters. They did not provide new messages in the last batched broadcast. Notably, since $t > \frac{n}{2}$, there will be at least three batched broadcasts.

Let's now consider a single round execution of these batched broadcasts. After each batched broadcast is executed, consider the predictor pair formed by the latest predictors of (1) the broadcasters of this batched broadcast; and (2) the broadcasters of the next batched broadcast. This predictor pair is attackable. We emphasize that the latest predictor of the "next" broadcasters can utilize not only the broadcast messages from this batched broadcast but also those they are going to broadcast in the next batched broadcast. For example, after $\{P_1, \ldots, P_{n-t}\}$ broadcast, $\{P_{n-t+1}, \ldots, P_{2(n-t)}\}$ as an entity, before making batched-broadcast, can already utilize the predictor based on the first $2(n-t)$ messages. This is crucial since this means that the latest predictor of the "next" broadcasters stays unchanged after their own broadcasting, which indicates that the above attackable pairs can form a chain.

However, this chain is not a valid one for arguing a non-negligible gap. This is because (1) it does not begin with an initial predictor; (2) it cannot be used directly if we consider $R > 1$ rounds. Unlike an unreactive functionality that leaves out $\{P_1, \ldots, P_{n-t}\}$ where these parties do not get any new messages after the functionality is executed, a batched broadcast from $\{P_1, \ldots, P_{n-t}\}$ will indeed also make these parties each get $n-t-1$ new correct messages[25]. In other words, after the first batched broadcast, the latest predictor of the broadcasters are not an initial predictor. We fix the above issues by adding a *dummy* batched broadcast from party $\{P_{n-t+1}, \ldots, P_{2(n-t)}\}$ at beginning of each round. Note that this is a dummy batched broadcast so the predictor of any parties would be unchanged after broadcasting. Essentially, $\{P_{n-t+1}, \ldots, P_{2(n-t)}\}$ is disjoint with the first broadcaster set and the last broadcaster set in a round, and so can be used to solder the chains of two consecutive rounds and, more importantly, solder the first chain to some initial predictor.

6.2.3 Resolving Challenges 1&2.

Consider a protocol in our (n, R, t)-unreactive syntax which uses both unreactive functionalities and a broadcast channel. Our above fixes imply that we can construct attackable predictor pair chains for the unreactive functionalities phase and the broadcast phase in each round. The remaining thing is to argue that we can glue all these chains. Recap that for the unreactive functionality fix, we rely on a Hamilton cycle in the Kneser graph $K(n, n-t)$. Since it is a cycle, we can use this cycle beginning from the vertex representing $\{P_{n-t+1}, \ldots, P_{2(n-t)}\}$. By doing so, all chains could be soldered together since (1) the set of parties not connecting to the last unreactive functionality in each round is disjoint with $\{P_{n-t+1}, \ldots, P_{2(n-t)}\}$, which is also the first *dummy* batched broadcaster

[25]Note this is not a problem if $t = n-1$.

in each round; and (2) the set of the last batched broadcasters is disjoint with $\{P_{n-t+1}, \ldots, P_{2(n-t)}\}$, which is also the set of parties not connecting to the first unreactive functionality in the next round.

6.2.4 Resolving the Special Case of $n = 5, t = 3$

Even though we do not have Hamilton cycle in $K(5, 2)$, we can add dummy unreactive functionalities into our syntax. Crucially, it suffices to find a path that traverses every vertex in the graph (perhaps more than once) from the vertex $\{P_{n-t+1}, \ldots, P_{2(n-t)}\}$ and goes back. It's well-known that on removing any vertex, the Petersen graph or $K(5, 2)$ is *Hamiltonian*. Thus, we can find such a path in $K(5, 2)$ of length 18.
To sum up, we have the following theorem.

Theorem 2. *For any n-party coin tossing protocol in the (n, t)-unreactive world, following the (n, R, t)-unreactive syntax, there exists a predict-and-quit adversarial strategy corrupting $\frac{n}{2} < t < n$ parties that can bias the output of some honest party by $\Omega\left(\frac{1}{\binom{n}{t}R}\right)$.*

7 Refinements: Cosmetic and Crucial

In this section, we show how to extend our lower bounds to coin tossing protocols that achieve just statistical agreement (as opposed to perfect like before). Additionally, we provide a general lower bound for coin tossing protocols in unreactive worlds where the bias induced by our adversary depends only on the *actual* number of hybrid functionalities invoked during the protocol, rather than the potentially much larger $\binom{n}{t}R$.

7.1 Ruling out Statistical Agreement

For ease of presentation, throughout the paper, we focused on coin tossing protocol with perfect agreement. Our proof technique (i.e., Sections 4 to 6) can be naturally extended to the statistical agreement setting, where the outputs of any two honest parties will differ with probability at most $\delta(\lambda)$ for some negligible function $\delta(\lambda)$. This can be done by applying a "Substitution Lemma" (Cf. Lemma 10) that allows replacement of random variables that are negligibly close.

Lemma 10 (Event Substitution)**.** *Consider three one-bit random variables A, B, C. Suppose $\Pr[B = C] \geq 1 - \delta(\lambda)$ for some negligible function $\delta(\lambda)$. Then*

$$|\Pr[A = B] - \Pr[A = C]| \leq \delta^*(\lambda)$$

for some negligible function $\delta(\lambda)$. We say that B can be substituted by C (and vice versa) in an equality event with negligible loss.

Proof. We have

$$\begin{aligned}
\Pr[A = B] &= \Pr[A = B \wedge B = C] + \Pr[A = B \wedge B \neq C] \\
&= \Pr[A = C \wedge B = C] + \Pr[A \neq C \wedge B \neq C] \\
&= \Pr[A = C | B = C] \Pr[B = C] + \Pr[A \neq C | B \neq C] \Pr[B \neq C]
\end{aligned}$$

Let $\Pr[A = C|B = C] = \beta_1$ and $\Pr[A \neq C|B \neq C] = \beta_2$. We have

$$\Pr[A = B] = \beta_1 \Pr[B = C] + \beta_2 \Pr[B \neq C]$$

We have

$$\begin{aligned}\Pr[A = C] &= \Pr[A = C \wedge A = B] + \Pr[A = C \wedge A \neq B] \\ &= \Pr[A = C \wedge B = C] + \Pr[A = C \wedge B \neq C] \\ &= \Pr[A = C|B = C]\Pr[B = C] + \Pr[A = C|B \neq C]\Pr[B \neq C]\end{aligned}$$

Let $\Pr[A = C|B \neq C] = \beta_3$. We have

$$\Pr[A = C] = \beta_1 \Pr[B = C] + \beta_3 \Pr[B \neq C]$$

Therefore,

$$|\Pr[A = B] - \Pr[A = C]| = \Pr[B \neq C]|\beta_2 - \beta_3| \leq \delta(\lambda)|\beta_2 - \beta_3| = \delta^*(\lambda)$$

for some negligible function $\delta(\lambda)$.

□

By Lemma 10, if two random variables disagree with negligible probability, we can safely substitute one by the other with negligible probability loss. Now, consider coin tossing protocols with statistical agreements. We can extend the notion of predictabilities of a party P_i to the probability that the output of the corresponding predictor equals the output of P_i (denoted by res_i). Since the outputs of any two parties should disagree with negligible probability, we can substitute any res_i by res_j with negligible probability loss. This lets us extend all of our analyses in a natural way to rule out coin tossing protocols with statistical agreement as well.

7.2 Relaxing the Need to Follow the Unreactive Syntax

Our theorems (Cf. Theorems 1 and 2) consider coin tossing protocols following the unreactive syntax. This is general since we can always compile a protocol in an unreactive world to one that follows the corresponding unreactive syntax (Cf. Lemma 2). However, this compilation will result in a blow-up in the number of rounds, which impacts the bias induced by our predict-and-quit adversaries. Note that the compilation only inserts *dummy* hybrid functionalities. Crucially, in an execution, the values of our predictabilities stay unchanged after a dummy hybrid functionality is invoked. That is, if there exists only F (non-dummy) invocations to hybrid functionalities in a coin tossing protocol, we should only need to include $\mathcal{O}(F)$ terms on the left-hand side of the triangle inequality (e.g., Equation (1)) which captures the sum of a chain of attackable pairs. This indicates the existence of an attackable predictor pair where their predictabilities have a gap of $\Omega(\frac{1}{F})$. Thus, we have the theorem as follows.

Theorem 3. *For any n-party coin tossing protocol in the (n, t)-unreactive world, with $\frac{n}{2} < t < n$, that involves at most F invocations of hybrid functionalities (unreactive or the broadcast), there exists a predict-and-quit adversarial strategy corrupting t parties that can bias the output of some honest party by $\Omega\left(\frac{1}{F}\right)$.*

References

[ABC$^+$85] Baruch Awerbuch, Manuel Blum, Benny Chor, Shafi Goldwasser, and Silvio Micali. How to implement bracha's o(log n) byzantine agreement algorithm. *Unpublished manuscript*, 1(2):10, 1985.

[ABDR04] Andris Ambainis, Harry Buhrman, Yevgeniy Dodis, and Hein Rohrig. Multiparty quantum coin flipping. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 250–259. IEEE, 2004.

[ABMO15] Gilad Asharov, Amos Beimel, Nikolaos Makriyannis, and Eran Omri. Complete characterization of fairness in secure two-party computation of boolean functions. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 199–228, 2015.

[Amb01] Andris Ambainis. A new protocol and lower bounds for quantum coin flipping. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 134–142, 2001.

[AN93] Noga Alon and Moni Naor. Coin-flipping games immune against linear-sized coalitions. *SIAM Journal on Computing*, 22(2):403–417, 1993.

[AO16] Bar Alon and Eran Omri. Almost-optimally fair multiparty coin-tossing with nearly three-quarters malicious. In *Theory of Cryptography Conference*, pages 307–335. Springer, 2016.

[Ash14] Gilad Asharov. Towards characterizing complete fairness in secure two-party computation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 291–316, 2014.

[ATSVY00] Dorit Aharonov, Amnon Ta-Shma, Umesh V Vazirani, and Andrew C Yao. Quantum bit escrow. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 705–714, 2000.

[BC90] Gilles Brassard and Claude Crépeau. Quantum bit commitment and coin tossing protocols. In *Conference on the Theory and Application of Cryptography*, pages 49–61. Springer, 1990.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[BHLT17] Niv Buchbinder, Iftach Haitner, Nissan Levi, and Eliad Tsfadia. Fair coin flipping: Tighter analysis and the many-party case. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2580–2600. SIAM, 2017.

[BHMO22] Amos Beimel, Iftach Haitner, Nikolaos Makriyannis, and Eran Omri. Tighter bounds on multiparty coin flipping via augmented weak martingales and differentially private sampling. *SIAM Journal on Computing*, 51(4):1126–1171, 2022.

[BHT18] Itay Berman, Iftach Haitner, and Aris Tentes. Coin flipping of any constant bias implies one-way functions. *Journal of the ACM (JACM)*, 65(3):1–95, 2018.

[BK14] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 421–439, 2014.

[Blu83] Manuel Blum. How to exchange (secret) keys. *AcM Transactions on computer systems (Tocs)*, 1(2):175–193, 1983.

[BOL85] Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 408–416. IEEE, 1985.

[BOO10] Amos Beimel, Eran Omri, and Ilan Orlov. Protocols for multiparty coin toss with dishonest majority. In *Annual Cryptology Conference*, pages 538–557. Springer, 2010.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19, 1988.

[CGL+18] Kai-Min Chung, Yue Guo, Wei-Kai Lin, Rafael Pass, and Elaine Shi. Game theoretic notions of fairness in multi-party coin toss. In *Theory of Cryptography Conference*, pages 563–596. Springer, 2018.

[CI93] Richard Cleve and Russell Impagliazzo. Martingales, collective coin flipping and discrete control processes. *other words*, 1(5), 1993.

[CL17] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *J. Cryptology*, 30(4):1157–1186, 2017.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 364–369, 1986.

[Fei99] Uriel Feige. Noncryptographic selection protocols. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 142–152. IEEE, 1999.

[FGH+02] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam D. Smith. Detectable byzantine agreement secure against faulty majorities. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24, 2002*, pages 118–126, 2002.

[FGMO05] Matthias Fitzi, Juan A. Garay, Ueli M. Maurer, and Rafail Ostrovsky. Minimal complete primitives for secure multi-party computation. *J. Cryptology*, 18(1):37–61, 2005.

[FGMvR02] Matthias Fitzi, Nicolas Gisin, Ueli M. Maurer, and Oliver von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 482–501, 2002.

[GHKL11] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. *J. ACM*, 58(6):24:1–24:37, 2011.

[GIM+10] S. Dov Gordon, Yuval Ishai, Tal Moran, Rafail Ostrovsky, and Amit Sahai. On complete primitives for fairness. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 91–108, 2010.

[GK09] S. Dov Gordon and Jonathan Katz. Complete fairness in multi-party computation without an honest majority. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 19–35, 2009.

[GKP15] Shafi Goldwasser, Yael Tauman Kalai, and Sunoo Park. Adaptively secure coin-flipping, revisited. In *International Colloquium on Automata, Languages, and Programming*, pages 663–674. Springer, 2015.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

[HIK+19] Shai Halevi, Yuval Ishai, Eyal Kushilevitz, Nikolaos Makriyannis, and Tal Rabin. On fully secure mpc with solitary output. In *Theory of Cryptography: 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part I 17*, pages 312–340. Springer, 2019.

[HKH20] Iftach Haitner and Yonatan Karidi-Heller. A tight lower bound on adaptively secure full-information coin flip. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1268–1276. IEEE, 2020.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[HO14] Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. *SIAM Journal on Computing*, 43(2):389–409, 2014.

[HT14] Iftach Haitner and Eliad Tsfadia. An almost-optimally fair three-party coin-flipping protocol. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 408–416, 2014.

[HZ10] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 466–485. Springer, 2010.

[IKK+11] Yuval Ishai, Jonathan Katz, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On achieving the "best of both worlds" in secure multiparty computation. *SIAM journal on computing*, 40(1):122–141, 2011.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235. IEEE Computer Society, 1989.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31, 1988.

[KKR21] Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin flipping protocols. *Combinatorica*, 41(1):75–98, 2021.

[KRS20] Ranjit Kumaresan, Srinivasan Raghuraman, and Adam Sealfon. Synchronizable exchange. *IACR Cryptol. ePrint Arch.*, 2020:976, 2020.

[KVV16] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 406–417, 2016.

[Lov78] László Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[MM+22] Arturo Merino, Torsten Mütze, et al. Kneser graphs are hamiltonian. *arXiv preprint arXiv:2212.03918*, 2022.

[MN05] Tal Moran and Moni Naor. Basing cryptographic protocols on tamper-evident seals. In *International Colloquium on Automata, Languages, and Programming*, pages 285–297. Springer, 2005.

[MNS09] Tal Moran, Moni Naor, and Gil Segev. An optimally fair coin toss. In *Theory of Cryptography Conference*, pages 1–18. Springer, 2009.

[MPS10] Hemanta K Maji, Manoj Prabhakaran, and Amit Sahai. On the computational complexity of coin flipping. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 613–622. IEEE, 2010.

[MW20] Hemanta K Maji and Mingyuan Wang. Black-box use of one-way functions is useless for optimal fair coin-tossing. In *Annual International Cryptology Conference*, pages 593–617. Springer, 2020.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of cryptology*, 4(2):151–158, 1991.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[PST17] Rafael Pass, Elaine Shi, and Florian Tramèr. Formal abstractions for attested execution secure processors. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 260–289, 2017.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85, 1989.

[RZ01] Alexander Russell and David Zuckerman. Perfect information leader election in log* n+ o (1) rounds. *Journal of Computer and System Sciences*, 63(4):612–626, 2001.

[Sak89] Michael Saks. A robust noncryptographic protocol for collective coin flipping. *SIAM Journal on Discrete Mathematics*, 2(2):240–244, 1989.

[WAS22] Ke Wu, Gilad Asharov, and Elaine Shi. A complete characterization of game-theoretically fair, multi-party coin toss. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 120–149. Springer, 2022.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.